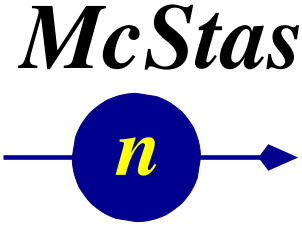
Physics Physics Department,
Technical University of Denmark
2800 Kongens Lyngby, Denmark



Component Manual for the Neutron Ray-Tracing Package McStas, version 3.5

P. Willendrup, E. Farhi, E. Knudsen, K. Lefmann

November, 2026

The software package McStas is a tool for carrying out Monte Carlo ray-tracing simulations of neutron scattering instruments with high complexity and precision. The simulations can compute all aspects of the performance of instruments and can thus be used to optimize the use of existing equipment, design new instrumentation, and carry out virtual experiments for e.g. training, experimental planning or data analysis. McStas is based on a unique design where an automatic compilation process translates high-level textual instrument descriptions into efficient ISO-C code. This design makes it simple to set up typical simulations and also gives essentially unlimited freedom to handle more unusual cases.

This report constitutes the reference manual for McStas, and, together with the manual for the McStas components, it contains documentation of most aspects of the program. It covers the various ways to compile and run simulations, a description of the meta-language used to define simulations, and some example simulations performed with the program.

This report documents McStas version 3.5, released November, 2026

The authors are:

Peter Kjær Willendrup <pkwi@fysik.dtu.dk>
Physics Department, Technical University of Denmark, Kongens Lyngby, Denmark

Emmanuel Farhi <emmanuel.farhi@synchrotron-soleil.fr>
Synchrotron SOLEIL, Saint-Aubin, France

Kim Lefmann <lefmann@nbi.dk>
Niels Bohr Institute, University of Copenhagen, Denmark

as well as authors who left the project:

Erik Knudsen <erkn@fysik.dtu.dk>
Physics Department, Technical University of Denmark, Kongens Lyngby, Denmark Jakob Garde <jaga@fysik.dtu.dk>
Physics Department, Technical University of Denmark, Kongens Lyngby, Denmark Peter Christiansen <pchristi@hep.lu.se>
Materials Research Department, Risø National Laboratory, Roskilde, Denmark
Present address: University of Lund, Lund, Sweden
Klaus Lieutenant <k.lieutenant@fz-juelich.dee>
Institut Laue-Langevin, Grenoble, France
Present address: Forschungs-Zentrum Jülich, Germany
Kristian Nielsen <kristian-nielsen@mail.tele.dk>
Materials Research Department, Risø National Laboratory, Roskilde, Denmark
Presently associated with: MySQL AB, Sweden

ISBN 978–87–550–3680–2
ISSN 0106–2840

Physics Department · DTU · 2026

Contents

Preface and acknowledgements **18**

1. About the component library **20**
1.1. Authorship . . . 20
1.2. Symbols for neutron scattering and simulation . . . 20
1.3. Component coordinate system . . . 21
1.4. About data files . . . 21
1.5. Component source code . . . 23
1.6. Documentation . . . 25
1.7. Component validation . . . 25
1.8. Disclaimer, bugs . . . 25

2. Monte Carlo Techniques and simulation strategy **27**
2.1. Neutron spectrometer simulations . . . 27
2.1.1. Monte Carlo ray tracing simulations . . . 27
2.2. The neutron weight . . . 28
2.2.1. Statistical errors of non-integer counts . . . 28
2.3. Weight factor transformations during a Monte Carlo choice . . . 29
2.3.1. Direction focusing . . . 30
2.4. Adaptive and Stratified sampling . . . 30
2.5. Accuracy of Monte Carlo simulations . . . 31

3. Source components **33**
3.0.1. Neutron flux . . . 33
3.1. The `Source_simple` McStas Component . . . 35
3.2. The `Source_div` McStas Component . . . 36
3.3. The `Source_Maxwell_3` McStas Component . . . 38
3.4. The `Source_gen` McStas Component . . . 39
3.5. The `Moderator` McStas Component . . . 44
3.6. The `Source_adapt` McStas Component . . . 46
3.7. The `Adapt_check` McStas Component . . . 50
3.8. The `Source_Optimizer` McStas Component . . . 52
3.9. The `Monitor_Optimizer` McStas Component . . . 56
3.10. Other sources components: contributed pulsed sources, virtual sources (event files) . . . 58

4. Beam optical components: Arms, slits, collimators, and filters **59**
4.1. The `Arm` McStas Component . . . 59

4.2. The `Slit` McStas Component . . . 60
4.3. The `Beamstop` McStas Component . . . 61
4.4. The `Filter_gen` McStas Component . . . 63
4.5. The `NCrystal_filter` McStas Component . . . 64
4.6. The `Collimator_linear` McStas Component . . . 67
4.7. The `Collimator_radial` McStas Component . . . 69

5. Reflecting optical components: mirrors, and guides 73
5.1. The `Guide` McStas Component . . . 73
5.2. The `Guide_gravity` McStas Component . . . 82

6. Moving optical components: Choppers and velocity selectors 85
6.1. The `DiskChopper` McStas Component . . . 85
6.2. The `FermiChopper` McStas Component . . . 88
6.3. The `Vitess_ChopperFermi` McStas Component . . . 95
6.4. The `V_selector` McStas Component . . . 100
6.5. The `Selector` McStas Component . . . 102

7. Polarising optics and magnetic field components 105
7.1. Setting and analysing polarisation . . . 105
7.2. The `Set_pol` McStas Component . . . 105
7.3. The `PolAnalyser_ideal` McStas Component . . . 106
7.4. The `Pol_SF_ideal` McStas Component . . . 107
7.5. Polarising mirrors and guides . . . 108
7.6. The `Pol_mirror` McStas Component . . . 108
7.7. The `Pol_bender` McStas Component . . . 109
7.8. The `Pol_guide_mirror` McStas Component . . . 111
7.9. The `Pol_guide_vmirror` McStas Component . . . 112
7.10. Magnetic field components . . . 115
7.11. The `Pol_Bfield` McStas Component . . . 115
7.12. The `Pol_Bfield_stop` McStas Component . . . 117
7.13. The `Pol_FieldBox` McStas Component . . . 118
7.14. The `Pol_constBfield` McStas Component . . . 119
7.15. The `Pol_tabled_field` McStas Component . . . 120

8. Monochromators 121
8.1. The `Monochromator_flat` McStas Component . . . 121
8.2. The `Monochromator_curved` McStas Component . . . 125
8.3. Monochromator_pol: A polarizing monochromator . . . 129
8.4. The `Monochromator_pol` McStas Component . . . 129
8.5. Single_crystal: Thick single crystal monochromator plate with multiple scattering . . . 131
8.6. Phase space transformer - moving monochromator . . . 131

9. Samples **132**
9.0.1. Neutron scattering notation . . . 133
9.0.2. Weight transformation in samples; focusing . . . 133
9.0.3. Small-angle scattering (SANS) and other sample models . . . 135
9.1. The `Incoherent` McStas Component . . . 135
9.1.1. Physics and algorithm . . . 139
9.1.2. Remark on functionality . . . 139
9.2. The `Tunneling_sample` McStas Component . . . 140
9.3. The `PowderN` McStas Component . . . 143
9.3.1. Files formats: powder structures . . . 148
9.3.2. Geometry, physical properties, concentricity . . . 149
9.3.3. Powder scattering . . . 149
9.3.4. Algorithm . . . 152
9.4. The `Single_crystal` McStas Component . . . 153
9.4.1. The physical model . . . 159
9.4.2. The algorithm . . . 162
9.4.3. Choosing the outgoing wave vector . . . 162
9.4.4. Computing the total coherent cross-section . . . 164
9.4.5. Implementation details . . . 165
9.5. The `Sans_spheres` McStas Component . . . 167
9.5.1. Small-angle scattering cross section . . . 168
9.5.2. Algorithm . . . 169
9.5.3. Calculating the weight factor . . . 169
9.6. The `Phonon_simple` McStas Component . . . 170
9.6.1. The phonon cross section . . . 172
9.6.2. The algorithm . . . 172
9.6.3. The weight transformation . . . 173
9.7. The `Isotropic_Sqw` McStas Component . . . 174
9.7.1. Neutron interaction with matter - overview . . . 181
9.7.2. Theoretical side . . . 182
9.7.3. Theoretical side - scattering in the sample . . . 183
9.7.4. The implementation . . . 189
9.7.5. Validation . . . 193
9.8. Magnetic samples, magnons and NCrystal . . . 195
9.9. The `Magnon_bcc` McStas Component . . . 195
9.10. The `Single_magnetic_crystal` McStas Component . . . 196
9.11. The `NCrystal_sample` McStas Component . . . 199
9.12. Samples for resolution-function calculations . . . 202
9.13. The `Res_sample` McStas Component . . . 202
9.14. The `TOFRes_sample` McStas Component . . . 203

10. Monitors and detectors **205**
10.1. The `TOF_monitor` McStas Component . . . 207
10.2. The `TOF2E_monitor` McStas Component . . . 208

10.3. The `E_monitor` McStas Component . . . 209
10.4. The `L_monitor` McStas Component . . . 210
10.5. The `PSD_monitor` McStas Component . . . 212
10.6. The `Divergence_monitor` McStas Component . . . 213
10.7. The `DivPos_monitor` McStas Component . . . 214
10.8. The `Monitor_nD` McStas Component . . . 217
10.9. The `Res_monitor` McStas Component . . . 231
10.10 Polarisation-sensitive monitors . . . 232
10.11 The `Pol_monitor` McStas Component . . . 232
10.12 The `PolLambda_monitor` McStas Component . . . 233
10.13 The `MeanPolLambda_monitor` McStas Component . . . 234
10.14 The `PSD_monitor_4PI_spin` McStas Component . . . 235

11. Special-purpose components **237**
11.1. MCPL: splitting simulations via portable neutron event files . . . 238
11.2. The `MCPL_input` McStas Component . . . 238
11.3. The `MCPL_output` McStas Component . . . 240
11.4. KDSource: kernel-density-estimated source from particle lists . . . 242
11.5. The `KDSource` McStas Component . . . 242
11.6. Text: drawing a text label in mcdisplay . . . 244
11.7. The `Text` McStas Component . . . 244
11.8. Progress_bar: Simulation progress and automatic saving . . . 245
11.9. Beam_spy: A beam analyzer . . . 245

12. The Union framework: sample environments and multiple scattering **246**
12.1. Union framework core: initialization, master, and termination . . . 246
12.2. The `Union_init` McStas Component . . . 246
12.3. The `Union_master` McStas Component . . . 247
12.4. The `Union_master_GPU` McStas Component . . . 248
12.5. The `Union_stop` McStas Component . . . 250
12.6. Union geometry components . . . 251
12.7. The `Union_box` McStas Component . . . 251
12.8. The `Union_cylinder` McStas Component . . . 253
12.9. The `Union_sphere` McStas Component . . . 255
12.10 The `Union_cone` McStas Component . . . 257
12.11 The `Union_mesh` McStas Component . . . 259
12.12 Union material definition . . . 262
12.13 The `Union_make_material` McStas Component . . . 262
12.14 The `Union_NCrystal_material` McStas Component . . . 263
12.15 Union scattering process components . . . 264
12.16 The `Non_process` McStas Component . . . 264
12.17 The `Incoherent_process` McStas Component . . . 266
12.18 The `Inhomogenous_incoherent_process` McStas Component . . . 267
12.19 The `IncoherentPhonon_process` McStas Component . . . 270

12.20 The PhononSimple_process McStas Component . . . 271
12.21 The Powder_process McStas Component . . . 273
12.22 The Single_crystal_process McStas Component . . . 274
12.23 The AF_HB_1D_process McStas Component . . . 277
12.24 The Texture_process McStas Component . . . 278
12.25 The NCrystal_process McStas Component . . . 280
12.26 The Template_process McStas Component . . . 281
12.27 The Template_surface McStas Component . . . 283
12.28 The Mirror_surface McStas Component . . . 284
12.29 Union loggers (event/history recording) . . . 285
12.30 The Union_logger_1D McStas Component . . . 285
12.31 The Union_logger_2DQ McStas Component . . . 287
12.32 The Union_logger_2D_kf McStas Component . . . 289
12.33 The Union_logger_2D_kf_time McStas Component . . . 291
12.34 The Union_logger_2D_space McStas Component . . . 294
12.35 The Union_logger_2D_space_time McStas Component . . . 296
12.36 The Union_logger_3D_space McStas Component . . . 298
12.37 Union absorption loggers . . . 300
12.38 The Union_abs_logger_1D_space McStas Component . . . 300
12.39 The Union_abs_logger_1D_space_event McStas Component . . . 302
12.40 The Union_abs_logger_1D_space_tof McStas Component . . . 304
12.41 The Union_abs_logger_1D_space_tof_to_lambda McStas Component . . . 306
12.42 The Union_abs_logger_1D_time McStas Component . . . 309
12.43 The Union_abs_logger_2D_space McStas Component . . . 311
12.44 The Union_abs_logger_event McStas Component . . . 314
12.45 The Union_abs_logger_nD McStas Component . . . 315
12.46 The Union_abs_logger_nD_scintillator McStas Component . . . 319
12.47 Union conditionals . . . 323
12.48 The Union_conditional_PSD McStas Component . . . 323
12.49 The Union_conditional_standard McStas Component . . . 325

13. SasView/sasmodels form factors: small-angle scattering 327
13.1. SasView form-factor components . . . 327
13.2. The SasView_adsorbed_layer McStas Component . . . 327
13.3. The SasView_barbell McStas Component . . . 329
13.4. The SasView_barbell_aniso McStas Component . . . 331
13.5. The SasView_bcc_paracrystal McStas Component . . . 332
13.6. The SasView_bcc_paracrystal_aniso McStas Component . . . 334
13.7. The SasView_binary_hard_sphere McStas Component . . . 336
13.8. The SasView_broad_peak McStas Component . . . 338
13.9. The SasView_capped_cylinder McStas Component . . . 339
13.10 The SasView_capped_cylinder_aniso McStas Component . . . 341
13.11 The SasView_core_multi_shell McStas Component . . . 343
13.12 The SasView_core_shell_bicelle McStas Component . . . 344

13.13 The SasView_core_shell_bicelle_aniso McStas Component 346
13.14 The SasView_core_shell_bicelle_elliptical McStas Component . . . 348
13.15 The SasView_core_shell_bicelle_elliptical_aniso McStas Component 350
13.16 The SasView_core_shell_bicelle_elliptical_belt_rough McStas Component 352
13.17 The SasView_core_shell_bicelle_elliptical_belt_rough_aniso McStas Component 354
13.18 The SasView_core_shell_cylinder McStas Component 357
13.19 The SasView_core_shell_cylinder_aniso McStas Component 359
13.20 The SasView_core_shell_ellipsoid McStas Component 361
13.21 The SasView_core_shell_ellipsoid_aniso McStas Component 362
13.22 The SasView_core_shell_parallelepiped McStas Component 364
13.23 The SasView_core_shell_parallelepiped_aniso McStas Component . . 367
13.24 The SasView_core_shell_sphere McStas Component 369
13.25 The SasView_correlation_length McStas Component 371
13.26 The SasView_cylinder McStas Component 373
13.27 The SasView_cylinder_aniso McStas Component 374
13.28 The SasView_dab McStas Component 376
13.29 The SasView_ellipsoid McStas Component 377
13.30 The SasView_ellipsoid_aniso McStas Component 379
13.31 The SasView_elliptical_cylinder McStas Component 381
13.32 The SasView_elliptical_cylinder_aniso McStas Component 383
13.33 The SasView_fcc_paracrystal McStas Component 385
13.34 The SasView_fcc_paracrystal_aniso McStas Component 386
13.35 The SasView_flexible_cylinder McStas Component 388
13.36 The SasView_flexible_cylinder_elliptical McStas Component 390
13.37 The SasView_fractal McStas Component 392
13.38 The SasView_fractal_core_shell McStas Component 393
13.39 The SasView_fuzzy_sphere McStas Component 395
13.40 The SasView_gauss_lorentz_gel McStas Component 397
13.41 The SasView_gaussian_peak McStas Component 398
13.42 The SasView_gel_fit McStas Component 400
13.43 The SasView_guinier McStas Component 401
13.44 The SasView_guinier_porod McStas Component 403
13.45 The SasView_hardsphere McStas Component 404
13.46 The SasView_hayter_msa McStas Component 405
13.47 The SasView_hollow_cylinder McStas Component 407
13.48 The SasView_hollow_cylinder_aniso McStas Component 409
13.49 The SasView_hollow_rectangular_prism McStas Component 411
13.50 The SasView_hollow_rectangular_prism_aniso McStas Component . . 413
13.51 The SasView_hollow_rectangular_prism_thin_walls McStas Component 415
13.52 The SasView_lamellar_hg McStas Component 416
13.53 The SasView_lamellar_hg_stack_caille McStas Component 418
13.54 The SasView_lamellar_stack_caille McStas Component 420

13.55 The SasView_lamellar_stack_paracrystal McStas Component 422
13.56 The SasView_line McStas Component 423
13.57 The SasView_linear_pearls McStas Component 425
13.58 The SasView_lorentz McStas Component 426
13.59 The SasView_mass_fractal McStas Component 428
13.60 The SasView_mass_surface_fractal McStas Component 429
13.61 The SasView_mono_gauss_coil McStas Component 431
13.62 The SasView_multilayer_vesicle McStas Component 432
13.63 The SasView_onion McStas Component 434
13.64 The SasView_parallelepiped McStas Component 435
13.65 The SasView_parallelepiped_aniso McStas Component 436
13.66 The SasView_peak_lorentz McStas Component 439
13.67 The SasView_pearl_necklace McStas Component 440
13.68 The SasView_poly_gauss_coil McStas Component 442
13.69 The SasView_polymer_excl_volume McStas Component 443
13.70 The SasView_polymer_micelle McStas Component 445
13.71 The SasView_porod McStas Component 446
13.72 The SasView_power_law McStas Component 448
13.73 The SasView_pringle McStas Component 449
13.74 The SasView_raspberry McStas Component 451
13.75 The SasView_rectangular_prism McStas Component 453
13.76 The SasView_rectangular_prism_aniso McStas Component 454
13.77 The SasView_rpa McStas Component 455
13.78 The SasView_sc_paracrystal McStas Component 456
13.79 The SasView_sc_paracrystal_aniso McStas Component 458
13.80 The SasView_sphere McStas Component 460
13.81 The SasView_spinodal McStas Component 461
13.82 The SasView_squarewell McStas Component 462
13.83 The SasView_stacked_disks McStas Component 464
13.84 The SasView_stacked_disks_aniso McStas Component 466
13.85 The SasView_star_polymer McStas Component 468
13.86 The SasView_stickyhardsphere McStas Component 469
13.87 The SasView_superball McStas Component 471
13.88 The SasView_superball_aniso McStas Component 472
13.89 The SasView_surface_fractal McStas Component 474
13.90 The SasView_teubner_strey McStas Component 476
13.91 The SasView_triaxial_ellipsoid McStas Component 477
13.92 The SasView_triaxial_ellipsoid_aniso McStas Component 479
13.93 The SasView_two_lorentzian McStas Component 481
13.94 The SasView_two_power_law McStas Component 483
13.95 The SasView_vesicle McStas Component 484

14. Contributed components **487**
14.1. Contributed sources 487

14.2. The ISIS_moderator McStas Component . . . 487
14.3. The ViewModISIS McStas Component . . . 490
14.4. The SNS_source McStas Component . . . 493
14.5. The SNS_source_analytic McStas Component . . . 494
14.6. The Source_gen4 McStas Component . . . 495
14.7. The Source_multi_surfaces McStas Component . . . 499
14.8. The Source_custom McStas Component . . . 500
14.9. The Source_pulsed McStas Component . . . 504
14.10 Contributed guides and benders . . . 506
14.11 The Guide_curved McStas Component . . . 506
14.12 The Guide_four_side McStas Component . . . 507
14.13 The Guide_gravity_psd McStas Component . . . 512
14.14 The Guide_honeycomb McStas Component . . . 514
14.15 The Guide_m McStas Component . . . 516
14.16 The Guide_multichannel McStas Component . . . 518
14.17 The Vertical_Bender McStas Component . . . 519
14.18 The NMO McStas Component . . . 521
14.19 The Conics_EH McStas Component . . . 523
14.20 The Conics_HE McStas Component . . . 524
14.21 The Conics_PH McStas Component . . . 526
14.22 The Conics_PP McStas Component . . . 528
14.23 The FlatEllipse_finite_mirror McStas Component . . . 529
14.24 The Mirror_conic McStas Component . . . 530
14.25 The Commodus_I3 McStas Component . . . 532
14.26 Contributed choppers and collimators . . . 533
14.27 The FermiChopper_ILL McStas Component . . . 533
14.28 The Fermi_chop2a McStas Component . . . 534
14.29 The MultiDiskChopper McStas Component . . . 535
14.30 The StatisticalChopper McStas Component . . . 537
14.31 The StatisticalChopper_Monitor McStas Component . . . 539
14.32 The Vertical_TOa McStas Component . . . 539
14.33 The Collimator_ROC McStas Component . . . 540
14.34 The Exact_radial_coll McStas Component . . . 541
14.35 The CavitiesIn McStas Component . . . 542
14.36 The CavitiesOut McStas Component . . . 543
14.37 The multi_pipe McStas Component . . . 544
14.38 Contributed filters and windows . . . 545
14.39 The Al_window McStas Component . . . 545
14.40 The Filter_graphite McStas Component . . . 546
14.41 The Sapphire_Filter McStas Component . . . 547
14.42 Contributed lenses and mirrors . . . 548
14.43 The Lens McStas Component . . . 548
14.44 The Lens_simple McStas Component . . . 551
14.45 The Mirror_Curved_Bispectral McStas Component . . . 552

14.46 The Mirror_Elliptic McStas Component . . . 553
14.47 The Mirror_Elliptic_Bispectral McStas Component . . . 554
14.48 The Mirror_Parabolic McStas Component . . . 555
14.49 The SupermirrorFlat McStas Component . . . 556
14.50 Contributed monochromators and analysers . . . 560
14.51 The Monochromator_2foc McStas Component . . . 560
14.52 The Monochromator_bent McStas Component . . . 562
14.53 The Monochromator_bent_complex McStas Component . . . 564
14.54 The PerfectCrystal McStas Component . . . 566
14.55 The Spherical_Backscattering_Analyser McStas Component . . . 569
14.56 Contributed polarisation components . . . 570
14.57 The Foil_flipper_magnet McStas Component . . . 570
14.58 The Pol_bender_tapering McStas Component . . . 571
14.59 The Pol_triafield McStas Component . . . 573
14.60 The Pol_pi_2_rotator McStas Component . . . 574
14.61 The Transmission_polarisatorABSnT McStas Component . . . 575
14.62 The Transmission_V_polarisator McStas Component . . . 577
14.63 The Spin_random McStas Component . . . 578
14.64 The PSD_Pol_monitor McStas Component . . . 579
14.65 The PSD_spinDmon McStas Component . . . 580
14.66 The PSD_spinUmon McStas Component . . . 581
14.67 Contributed single-crystal and powder/polycrystalline samples . . . 582
14.68 The Single_crystal_inelastic McStas Component . . . 582
14.69 The SiC McStas Component . . . 585
14.70 The Spot_sample McStas Component . . . 586
14.71 The GISANS_sample McStas Component . . . 587
14.72 Contributed inelastic (phonon and magnon) samples . . . 590
14.73 The Dispersion_relation McStas Component . . . 590
14.74 The Phonon_BvK_PG McStas Component . . . 593
14.75 The SpinWave_BCO McStas Component . . . 595
14.76 Contributed SANS samples . . . 597
14.77 The SANSCurve McStas Component . . . 597
14.78 The SANSCylinders McStas Component . . . 598
14.79 The SANSEllipticCylinders McStas Component . . . 599
14.80 The SANSLiposomes McStas Component . . . 600
14.81 The SANSNanodiscs McStas Component . . . 601
14.82 The SANSNanodiscsFast McStas Component . . . 603
14.83 The SANSNanodiscsWithTags McStas Component . . . 604
14.84 The SANSNanodiscsWithTagsFast McStas Component . . . 606
14.85 The SANSPDB McStas Component . . . 608
14.86 The SANSPDBFast McStas Component . . . 609
14.87 The SANSShells McStas Component . . . 610
14.88 The SANSSpheres McStas Component . . . 611
14.89 The SANSSpheresPolydisperse McStas Component . . . 612

14.90 The SANS_AnySamp McStas Component . . . 613
14.91 The SANS_DebyeS McStas Component . . . 614
14.92 The SANS_Guinier McStas Component . . . 615
14.93 The SANS_Liposomes_Abs McStas Component . . . 616
14.94 The SANS_benchmark2 McStas Component . . . 618
14.95 The Sans_liposomes_new McStas Component . . . 619
14.96 The Multilayer_Sample McStas Component . . . 620
14.97 Contributed monitors and detectors . . . 622
14.98 The E_4PI McStas Component . . . 622
14.99 The NPI_tof_dhkl_detector McStas Component . . . 623
14.100 The NPI_tof_theta_monitor McStas Component . . . 625
14.101 The PSD_Detector McStas Component . . . 626
14.102 The PSD_monitor_Filter McStas Component . . . 630
14.103 The PSD_monitor_rad McStas Component . . . 631
14.104 The Radial_div McStas Component . . . 632
14.105 The SANSQMonitor McStas Component . . . 633
14.106 The TOFSANSdet McStas Component . . . 634
14.107 The TOF2Q_cyl_monitor McStas Component . . . 635
14.108 The TOF_PSDmonitor McStas Component . . . 636
14.109 The TOF_PSDmonitor_toQ McStas Component . . . 637
14.110 Other contributed components . . . 638
14.111 The Wavelength_Diffuser McStas Component . . . 638
14.112 Contributed MCViNE sample components . . . 639
14.113 The MCViNE_SQ McStas Component . . . 639
14.114 The MCViNE_SvQ McStas Component . . . 641
14.115 The MCViNE_SQE McStas Component . . . 643
14.116 The MCViNE_E_Q McStas Component . . . 645
14.117 The MCViNE_E_vQ McStas Component . . . 647
14.118 The MCViNE_Broadened_E_Q McStas Component . . . 649
14.119 The MCViNE_LorentzianBroadened_E_Q McStas Component . . . 651
14.120 The MCViNE_ConstantEnergyTransfer McStas Component . . . 652
14.121 The MCViNE_ConstantQE McStas Component . . . 654
14.122 The MCViNE_ConstantvQE McStas Component . . . 656
14.123 The MCViNE_Phonon_IncoherentElastic McStas Component . . . 657
14.124 The MCViNE_Phonon_IncoherentInelastic McStas Component . . . 659
14.125 The MCViNE_Phonon_CoherentInelastic_PolyXtal McStas Component . . . 661
14.126 The MCViNE_Phonon_CoherentInelastic_SingleXtal McStas Component 663
14.127 The MCViNE_DGSSXRes McStas Component . . . 665
14.128 The MCViNE_SANS2D_ongrid McStas Component . . . 667
14.129 Contributed MCViNE kernels as Union processes . . . 668
14.130 The MCViNE_SQ_process McStas Component . . . 669
14.131 The MCViNE_SvQ_process McStas Component . . . 670
14.132 The MCViNE_SQE_process McStas Component . . . 672
14.133 The MCViNE_E_Q_process McStas Component . . . 673

14.134 The MCViNE_E_vQ_process McStas Component . . . 675
14.135 The MCViNE_Broadened_E_Q_process McStas Component . . . 677
14.136 The MCViNE_LorentzianBroadened_E_Q_process McStas Component . . 678
14.137 The MCViNE_ConstantEnergyTransfer_process McStas Component . . . 680
14.138 The MCViNE_ConstantQE_process McStas Component . . . 681
14.139 The MCViNE_ConstantvQE_process McStas Component . . . 682
14.140 The MCViNE_Phonon_IncoherentElastic_process McStas Component . . 684
14.141 The MCViNE_Phonon_IncoherentInelastic_process McStas Component 685
14.142 The MCViNE_Phonon_CoherentInelastic_PolyXtal_process McStas Component . . . 687
14.143 The MCViNE_Phonon_CoherentInelastic_SingleXtal_process McStas Component . . . 689
14.144 The MCViNE_DGSSXRes_process McStas Component . . . 691
14.145 The MCViNE_SANS2D_ongrid_process McStas Component . . . 692

15. Obsolete components **694**
15.1. The V_sample McStas Component . . . 694
15.2. The Virtual_input McStas Component . . . 696
15.3. The Virtual_output McStas Component . . . 698
15.4. The Virtual_mcnp_input McStas Component . . . 699
15.5. The Virtual_mcnp_output McStas Component . . . 701
15.6. The Virtual_mcnp_ss_input McStas Component . . . 702
15.7. The Virtual_mcnp_ss_output McStas Component . . . 703
15.8. The Virtual_mcnp_ss_Guide McStas Component . . . 705
15.9. The Virtual_tripoli4_input McStas Component . . . 706
15.10. The Virtual_tripoli4_output McStas Component . . . 708
15.11. The Vitess_input McStas Component . . . 709
15.12. The Vitess_output McStas Component . . . 710
15.13. The ESS_moderator_short McStas Component . . . 711
15.14. The ESS_moderator McStas Component . . . 713
15.15. The Beam_spy McStas Component . . . 715
15.16. The SasView_model McStas Component . . . 716
15.17. The Guide_anyshape_r McStas Component . . . 735
15.18. The MCPL_input_once McStas Component . . . 737

A. Polarisation in McStas **739**
A.1. The Polarization Vector . . . 739
A.1.1. Example: Magnetic fields . . . 741
A.2. Polarized Neutron Scattering . . . 742
A.2.1. Example: Nuclear scattering . . . 743
A.2.2. Example: Polarizing Monochromator and Guides . . . 744
A.3. Magnetic fields: the precession algorithm and field framework . . . 746
A.3.1. The magnetic field stack . . . 746
A.3.2. Numerical spin precession: SimpleNumMagnetPrecession . . . 747

A.3.3. Available field functions . . . 748
A.4. Polarisation components in McStas . . . 749
A.4.1. Setting and analysing polarisation . . . 749
A.4.2. Polarising monochromators, mirrors and guides . . . 750
A.4.3. Magnetic field components . . . 750
A.4.4. Polarisation-sensitive monitors . . . 751
A.4.5. Samples and polarisation . . . 751
A.5. Test and example instruments . . . 752
A.5.1. Component-level regression tests . . . 752
A.5.2. Test_Pol_TripleAxis . . . 752
A.5.3. He3_spin_filter . . . 752
A.5.4. A spin-echo example: SE_example . . . 752

B. The Union components **754**
B.1. Motivation . . . 754
B.1.1. Volumes, priority and overlap . . . 755
B.2. Component classes . . . 755
B.2.1. Loading the Union library . . . 756
B.2.2. Scattering processes . . . 756
B.2.3. Union_make_material . . . 758
B.2.4. Geometry components . . . 759
B.2.5. Union_master . . . 761
B.2.6. Surfaces . . . 762
B.2.7. Logger components . . . 762
B.2.8. Conditional components . . . 763
B.2.9. Tagging: ray histories . . . 763
B.3. The propagation algorithm . . . 764
B.3.1. Basic Monte Carlo step . . . 764
B.3.2. Intersection and destination lists . . . 764
B.3.3. Known limitations . . . 765
B.4. A complete example . . . 766
B.5. Further examples and worked tutorials . . . 767

C. Libraries and constants **768**
C.1. Run-time calls and functions (`mcstas-r`) . . . 768
C.1.1. Neutron propagation . . . 768
C.1.2. Coordinate and component variable retrieval . . . 769
C.1.3. Coordinate transformations . . . 771
C.1.4. Mathematical routines . . . 771
C.1.5. Output from detectors . . . 772
C.1.6. Ray-geometry intersections . . . 772
C.1.7. Random numbers . . . 772
C.2. Reading a data file into a vector/matrix (Table input, `read_table-lib`) . . . 773
C.3. Monitor_nD Library . . . 776

C.4. Adaptive importance sampling Library 776
C.5. Vitess import/export Library 776
C.6. Constants for unit conversion etc. 776

D. The McStas terminology **778**

Bibliography **779**

Index and keywords **781**

Preface and acknowledgements

This document contains information on the neutron scattering components which are the building blocks for defining instruments in the Monte Carlo neutron ray-tracing program McStas version 3.5. The initial release in October 1998 of version 1.0 was presented in Ref. [LN99] and further developed though version 2.0 as presented in Ref. [Wil+14]. The reader of this document is not supposed to have specific knowledge of neutron scattering, but some basic understanding of physics is helpful in understanding the theoretical background for the component functionality. For details about setting up and running simulations, we refer to the McStas system manual [Wil+05]. We assume familiarity with the use of the C programming language.

It is a pleasure to thank Dir. Kurt N. Clausen, PSI, for his continuous support to McStas and for having initiated the project. Continuous support to McStas has also come from Prof. Robert McGreevy, ISIS. Apart from the authors of this manual, also Per-Olof Åstrand, NTNU Trondheim, has contributed to the development of the McStas system. We have further benefited from discussions with many other people in the neutron scattering community, too numerous to mention here.

The users who contributed components to this manual are acknowledged as authors of the individual components. We encourage other users to contribute components with manual entries for inclusion in future versions of McStas.

In case of errors, questions, or suggestions, do not hesitate to contact the authors at `mcstas@risoe.dk` or consult the McStas home page [Mcsa]. A special bug/request reporting service is available [Git].

We would like to kindly thank all McStas component contributors. This is the way we improve the software alltogether.

The McStas project has been supported by the European Union through"XENNI / Cool Neutrons" (FP4), "SCANS" (FP5), "nmi3/MCNSI" (FP6), "'nmi3-ii/E-learning" and "nmi3-ii/MCNSI7" (FP7) [Nmi; Mcn]. McStas was supported directly from the construction project for the ISIS second target station (TS2/EU), see [Ts2]. Currently McStas is supported through the Danish involvement in the *Data Management and Software Center*, a subdivision of the European Spallation Source (ESS), see [Ess] and the European Union through "SINE2020/WP3 e-elarning" and and "SINE2020/WP8 e-Tools" (Horizon2020). the home pages [Sin].

If you **appreciate** this software, please subscribe to the `neutron-mc@risoe.dk` email list, send us a smiley message, and contribute to the package. We also encourage you to refer to this software when publishing results, with the following citations:

- P. Willendrup, E. Farhi, E. Knudsen, U. Filges and K. Lefmann, Journal of Neutron Research, **17** (2014) 35.

- K. Lefmann and K. Nielsen, Neutron News **10/3**, 20, (1999).
- P. Willendrup, E. Farhi and K. Lefmann, Physica B, **350** (2004) 735.

1. About the component library

This McStas Component Manual consists of the following major parts:

- An introduction to the use of Monte Carlo methods in McStas.
- A thorough description of system components, with one chapter per major category: Sources, optics, monochromators, samples, monitors, and other components.
- The McStas library functions and definitions that aid in the writing of simulations and components in Appendix C.
- An explanation of the McStas terminology in Appendix D.

Additionally, you may refer to the list of example instruments from the library in the McStas User Manual.

1.1. Authorship

The component library is maintained by the McStas system group. A number of basic components "belongs" the McStas system, and are supported and tested by the McStas team.

Other components are contributed by specific authors, who are listed in the code for each component they contribute as well as in this manual. McStas users are encouraged to send their contributions to us for inclusion in future releases.

Some contributed components have later been taken over for further development by the McStas system group, with permission from the original authors. The original authors will still appear both in the component code and in the McStas manual.

1.2. Symbols for neutron scattering and simulation

In the description of the theory behind the component functionality we will use the usual symbols $\mathbf{r}$ for the position (x, y, z) of the particle (unit m), and $\mathbf{v}$ for the particle velocity (v_x, v_y, v_z) (unit m/s). Another essential quantity is the neutron wave vector $\mathbf{k} = m_\mathrm{n}\mathbf{v}/\hbar$, where m_n is the neutron mass. $\mathbf{k}$ is usually given in $\mathrm{Å}^{-1}$, while neutron energies are given in meV. The neutron wavelength is the reciprocal wave vector, $\lambda = 2\pi/k$. In general, vectors are denoted by boldface symbols.

Subscripts "i" and "f" denotes "initial" and "final", respectively, and are used in connection with the neutron state before and after an interaction with the component in question.

The spin of the neutron is given a special treatment. Despite the fact that each physical neutron has a well defined spin value, the McStas spin vector **s** can have any length between zero (unpolarized beam) and unity (totally polarized beam). Further, all three cartesian components of the spin vector are present simultaneously, although this is physically not permitted by quantum mechanics. For further details about polarization handling, you may refer to the Polarisation chapter, A.

1.3. Component coordinate system

All mentioning of component geometry refer to the local coordinate system of the individual component. The axis convention is so that the z axis is along the neutron propagation axis, the y axis is vertical up, and the x axis points left when looking along the z-axis, completing a right-handed coordinate system. Most components 'position' (as specified in the instrument description with the `AT` keyword) corresponds to their input side at the nominal beam position. However, a few components are radial and thus positioned in their centre.

Components are usually not designed to overlap. This may lead to loss of neutron rays. Warnings will be issued during simulation if sections of the instrument are not reached by any neutron rays, or if neutrons are removed. This is usually the sign of either overlapping components or a very low intensity.

1.4. About data files

Some components require external data files, e.g. lattice crystallographic definitions for Laue and powder pattern diffraction, $S(q,\omega)$ tables for inelastic scattering, neutron events files for virtual sources, transmission and reflectivity files, etc.

Such files distributed with McStas are located in the `data` sub-directory of the McStas library. Components that make use of the McStas file system, including the `read-table` library (see section C.2) may access all McStas data files without making local copies. Of course, you are welcome to define your own data files, and eventually contribute to McStas if you find them useful.

File extensions are not compulsory but help in identifying relevant files per application. We list powder and liquid data files from the McStas library in Tables 1.2 and 1.3. These files contain an extensive header describing physical properties with references, and are specially suited for the PowderN (see 9.3) and Isotropic_Sqw components (see 9.7). Whenever using any $S(q,\omega)$ data for the Isotropic_Sqw component, we kindly request to cite the corresponding reference.

MCSTAS/data	Description
*.lau	Laue pattern file, as issued from Crystallographica. For use with Single_crystal, PowderN, and Isotropic_Sqw. Data: [h k l Mult. d-space 2Theta F-squared]
*.laz	Powder pattern file, as obtained from Lazy/ICSD. For use with PowderN, Isotropic_Sqw and possibly Single_crystal.
*.trm	transmission file, typically for monochromator crystals and filters. Data: [k (Angs-1) , Transmission (0-1)]
*.rfl	reflectivity file, typically for mirrors and monochromator crystals. Data: [k (Angs-1) , Reflectivity (0-1)]
*.sqw	$S(q,\omega)$ files for Isotropic_Sqw component. Data: [q] [ω] [$S(q,\omega)$]

Table 1.1.: Data files of the McStas library.

Table 1.3.: Liquids of the McStas library [Ics; DL03]. Low c and high σ_{abs} materials are highlighted.

MCSTAS/data File name	σ_{coh} [barns]	σ_{inc} [barns]	σ_{abs} [barns]	T_m [K]	c [m/s]	Note
Cs_liq_tot.sqw	3.69	0.21	29.0	301.6	**1090**	liquid cesium. Bodensteiner et al, Phys. Rev. A, 45 (1992) 5709 and 46 (1992) 3574.
D2_liq_21_tot.sqw	7.64	0	0.00052	21		liquid deuterium. Measured. Frei et al, PHYSICAL REVIEW B 80 (2009) 064301
D2_pow_21_tot.sqw	7.64	0	0.00052	12		powder deuterium. Measured. Frei et al, PHYSICAL REVIEW B 80 (2009) 064301
D2O_liq_290_coh and D_liq_290_inc	15.4	4.1	0.0012	290		liquid heavy water. Classical MD. E. Farhi et al, J. Nucl. Sci. Tech. (2015)
D2O_liq_290_tot	15.4	4.1	0.0012	290		liquid heavy water. Measured. E. Farhi et al, J. Nucl. Sci. Tech. (2015)
Ge_liq_coh.sqw and Ge_liq_inc.sqw	8.42	0.18	2.2	1211.4	5400	liquid germanium. Ab-initio MD. Hugouvieux V, Farhi E, Johnson MR, et al., PRB 75 (2007) 104208

(continued on next page)

(Table 1.3 continued from previous page)

MCSTAS/data File name	σ_{coh} [barns]	σ_{inc} [barns]	σ_{abs} [barns]	T_m [K]	c [m/s]	Note
H2O_liq_290_coh and H2O_liq_290_inc	7.75	161	0.665	290	1530	liquid water. Classical MD. E. Farhi et al, J. Nucl. Sci. Tech. (2015)
H2O_liq_290_tot	7.75	161	0.665	290	1530	liquid water. Measured. E. Farhi et al, J. Nucl. Sci. Tech. (2015)
He4_liq_coh.sqw	1.34	0	0.00747	2	**240**	liquid helium. Measured (constant width). R.J. Donnelly et al., J. Low Temp. Phys., 44 (1981) 471
He4_liq_1_coh.sqw	1.34	0	0.00747	1	**240**	liquid helium. Measured. K. Andersen et al, J Phys Cond Mat, 6 (1994) 821
Ne_liq_tot.sqw	2.62	0.008	0.039	24.56	**591**	liquid neon. Measured. Buyers, Sears et al, Phys Rev A 11 (1975) 697
Rb_liq_coh.sqw and Rb_liq_inc.sqw	6.32	0.5	0.38	312.46	**1300**	liquid rubidium. Classical MD. E. Farhi, V. Hugouvieux, M.R. Johnson, W. Kob, Journal of Computational Physics 228 (2009) 5251-5261
Rb_liq_tot.sqw	6.32	0.5	0.38	312.46	**1300**	liquid rubidium. Measured. J. Copley et al, PRA 9 (1974) 1656 and PRL 32 (1974) 49

McStas itself generates both simulation and monitor data files, which structure is explained in the User Manual (see end of chapter 'Running McStas').

1.5. Component source code

Source code for all components may be found in the `MCSTAS` library subdirectory of the McStas installation; the recommended installation route is via the `conda-forge` channel (see the User Manual, section on installation), in which case the library lives under the conda environment's resource tree. The library location may be queried with `mcrun --showcfg=resourcedir`, or overridden using the `MCSTAS` environment variable.

MCSTAS/data File name	σ_{coh} [barns]	σ_{inc} [barns]	σ_{abs} [barns]	T_m [K]	c [m/s]	Note
Ag.laz	4.407	0.58	**63.3**	1234.9	2600	
Al2O3_sapphire.laz	15.683	0.0188	0.4625	2273		
Al.laz	1.495	0.0082	0.231	933.5	5100	.lau
Au.laz	7.32	0.43	**98.65**	1337.4	**1740**	
B4C.laz	19.71	6.801	**3068**	2718		
Ba.laz	3.23	0.15	29.0	1000	**1620**	
Be.laz	7.63	0.0018	0.0076	1560	13000	
BeO.laz	11.85	0.003	0.008	2650		.lau
Bi.laz	9.148	0.0084	0.0338	544.5	**1790**	
C60.lau	5.551	0.001	0.0035			
C_diamond.laz	5.551	0.001	0.0035	4400	18350	.lau
C_graphite.laz	5.551	0.001	0.0035	3800	18350	.lau
Cd.laz	3.04	3.46	**2520**	594.2	2310	
Cr.laz	1.660	1.83	3.05	2180	5940	
Cs.laz	3.69	0.21	29.0	301.6	**1090**	c in liquid
Cu.laz	7.485	0.55	3.78	1357.8	3570	
Fe.laz	11.22	0.4	2.56	1811	4910	
Ga.laz	6.675	0.16	2.75	302.91	2740	
Gd.laz	29.3	151	**49700**	1585	2680	
Ge.laz	8.42	0.18	2.2	1211.4	5400	
H2O_ice_1h.laz	7.75	160.52	0.6652	273		
Hg.laz	20.24	6.6	**372.3**	234.32	**1407**	
I2.laz	7.0	0.62	12.3	386.85		
In.laz	2.08	0.54	**193.8**	429.75	**1215**	
K.laz	.69	0.27	2.1	336.53	**2000**	
LiF.laz	4.46	0.921	**70.51**	1140		
Li.laz	0.454	0.92	**70.5**	453.69	6000	
Nb.laz	8.57	0.0024	1.15	2750	3480	
Ni.laz	13.3	5.2	4.49	1728	4970	
Pb.laz	11.115	0.003	0.171	600.61	**1260**	
Pd.laz	4.39	0.093	6.9	1828.05	3070	
Pt.laz	11.58	0.13	10.3	2041.4	2680	
Rb.laz	6.32	0.5	0.38	312.46	**1300**	
Se_alpha.laz	7.98	0.32	11.7	494	3350	
Se_beta.laz	7.98	0.32	11.7	494	3350	
Si.laz	2.163	0.004	0.171	1687	2200	
SiO2_quartza.laz	10.625	0.0056	0.1714	846		.lau
SiO2_quartzb.laz	10.625	0.0056	0.1714	1140		.lau
Sn_alpha.laz	4.871	0.022	0.626	505.08		
Sn_beta.laz	4.871	0.022	0.626	505.08	2500	
Ti.laz	1.485	2.87	6.09	1941	4140	
Tl.laz	9.678	0.21	3.43	577	**818**	
V.laz	.0184	4.935	5.08	2183	4560	
Zn.laz	4.054	0.077	1.11	692.68	3700	
Zr.laz	6.44	0.02	0.185	2128	3800	

Table 1.2.: Powders of the McStas library [Ics; DL03]. Low c and high σ_{abs} materials are highlighted. Files are given in LAZY format, but may exist as well in Crystallographica *.lau* format as well.

In case users only require to add new features, preserving the existing features of a component, using the `EXTEND` keyword in the instrument description file is recommended. For larger modification of a component, it is advised to make a copy of the component file into the working directory. A component file in the local directory will in McStas take precedence over a library component of the same name.

1.6. Documentation

As a complement to this Component Manual, we encourage users to use the `mcdoc` front-end which enables to display both the catalog of the McStas library, e.g using:

```
1  mcdoc
```

as well as the documentation of specific components, e.g with:

```
1  mcdoc searchterm
2  mcdoc file.comp
```

The first line will search for all components matching *searchterm*, and open (or list, if `BROWSER` is unset) their documentation. For instance, `mcdoc laz` will list all available Lazy data files, whereas `mcdoc Monitor` will list most Monitors. The second example will display the help corresponding to the *file.comp* component, using your BROWSER setting, or as text if unset. The `--help` option will display the command help, as usual.

An overview of the component library is also given at the McStas home page [Mcsa] and in the User Manual [Wil+05].

1.7. Component validation

Components are continuously exercised by the `mctest`/`mcviewtest` regression suite (see the User Manual's section on self-testing the installation), which compiles and runs the example instrument library and compares results across platforms, CPU vs. GPU, and McStas versions using `mcplotdiff-html`. Some components additionally underwent dedicated, published validation against analytical models and/or measurements; velocity selectors and Fermi choppers were treated as black boxes and their line shapes cross-checked against analytical functions, and `Guide`/`Guide_gravity` trajectories were checked (with and without gravity) against analytically-computed reflection positions, directions and losses. `Source_gen` was cross-checked against an independent source module for a 3-Maxwellian moderator model, in wavelength/divergence line shape and absolute flux.

1.8. Disclaimer, bugs

We would like to emphasize that the usage of both the McStas software, as well as its components are the responsability of the users. Indeed, obtaining accurate and reliable results requires a substantial work when writing instrument descriptions. This also means that users should read carefully both the documentation from the manuals [Wil+05] and

from the component itself (using `mcdoc` *comp*) before reporting errors. Most anomalous results often originate from a wrong usage of some part of the package.

Anyway, if you find that either the documentation is not clear, or the behavior of the simulation is undoubtedly anomalous, you should report this to us at `mcstas@risoe.dk` and refer to our special bug/request reporting service [Git].

2. Monte Carlo Techniques and simulation strategy

This chapter explains the simulation strategy and the Monte Carlo techniques used in McStas. We first explain the concept of the neutron weight factor, and discuss the statistical errors in dealing with sums of neutron weights. Secondly, we give an expression for how the weight factor transforms under a Monte Carlo choice and specialize this to the concept of direction focusing. Finally, we present a way of generating random numbers with arbitrary distributions. More details are available in the Appendix concerning random numbers.

2.1. Neutron spectrometer simulations

2.1.1. Monte Carlo ray tracing simulations

The behavior of a neutron scattering instrument can in principle be described by a complex integral over all relevant parameters, like initial neutron energy and divergence, scattering vector and position in the sample, etc. However, in most relevant cases, these integrals are not solvable analytically, and we hence turn to Monte Carlo methods. The neutron ray-tracing Monte Carlo method has been used widely for guide studies [Cop93; Far+02; Sch+04], instrument optimization and design [ZLa04; Lie05]. Most of the time, the conclusions and general behavior of such studies may be obtained using the classical analytic approaches, but accurate estimates for the flux, resolution and generally the optimum parameter set, benefit considerably from MC methods.

Mathematically, the Monte-Carlo method is an application of the law of large numbers [Jam80; GRR92]. Let $f(u)$ be a finite continuous integrable function of parameter u for which an integral estimate is desirable. The discrete statistical mean value of f (computed as a series) in the uniformly sampled interval $a < u < b$ converges to the mathematical mean value of f over the same interval.

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1, a\le u_i\le b}^{n} f(u_i) = \frac{1}{b-a} \int_a^b f(u)du \tag{2.1}$$

In the case were the u_i values are regularly sampled, we come to the well known midpoint integration rule. In the case were the u_i values are randomly (but uniformly) sampled, this is the Monte-Carlo integration technique. As random generators are not perfect, we rather talk about *quasi*-Monte-Carlo technique. We encourage the reader to consult James [Jam80] for a detailed review on the Monte-Carlo method.

2.2. The neutron weight

A totally realistic semi-classical simulation will require that each neutron is at any time either present or lost. In many instruments, only a very small fraction of the initial neutrons will ever be detected, and simulations of this kind will therefore waste much time in dealing with neutrons that never hit the relevant detector or monitor.

An important way of speeding up calculations is to introduce a neutron "weight factor" for each simulated neutron ray and to adjust this weight according to the path of the ray. If *e.g.* the reflectivity of a certain optical component is 10%, and only reflected neutrons ray are considered later in the simulations, the neutron weight will be multiplied by 0.10 when passing this component, but every neutron is allowed to reflect in the component. In contrast, the totally realistic simulation of the component would require in average ten incoming neutrons for each reflected one.

Let the initial neutron weight be p_0 and let us denote the weight multiplication factor in the j'th component by π_j. The resulting weight factor for the neutron ray after passage of the n components in the instrument becomes the product of all contributions

$$p = p_n = p_0 \prod_{j=1}^{n} \pi_j. \tag{2.2}$$

Each adjustment factor should be $0 < \pi_j < 1$, except in special circumstances, so that total flux can only decrease through the simulation, see section 2.3. For convenience, the value of p is updated (within each component) during the simulation.

Simulation by weight adjustment is performed whenever possible. This includes

- Transmission through filters and windows.
- Transmission through Soller blade collimators and velocity selectors (in the approximation which does not take each blade into account).
- Reflection from monochromator (and analyzer) crystals with finite reflectivity and mosaicity.
- Reflection from guide walls.
- Passage of a continuous beam through a chopper.
- Scattering from all types of samples.

2.2.1. Statistical errors of non-integer counts

In a typical simulation, the result will consist of a count of neutrons histories ("rays") with different weights. The sum of these weights is an estimate of the mean number of neutrons hitting the monitor (or detector) per second in a “real” experiment. One may write the counting result as

$$I = \sum_i p_i = N\overline{p}, \tag{2.3}$$

where N is the number of rays hitting the detector and the horizontal bar denotes averaging. By performing the weight transformations, the (statistical) mean value of I is unchanged. However, N will in general be enhanced, and this will improve the accuracy of the simulation.

To give an estimate of the statistical error, we proceed as follows: Let us first for simplicity assume that all the counted neutron weights are almost equal, $p_i \approx \overline{p}$, and that we observe a large number of neutrons, $N \geq 10$. Then N almost follows a normal distribution with the uncertainty $\sigma(N) = \sqrt{N}$ [1]. Hence, the statistical uncertainty of the observed intensity becomes

$$\sigma(I) = \sqrt{N}\overline{p} = I/\sqrt{N}, \tag{2.4}$$

as is used in real neutron experiments (where $\overline{p} \equiv 1$). For a better approximation we return to Eq. (2.3). Allowing variations in both N and $\overline{p}$, we calculate the variance of the resulting intensity, assuming that the two variables are statistically independent:

$$\sigma^2(I) = \sigma^2(N)\overline{p}^2 + N^2\sigma^2(\overline{p}). \tag{2.5}$$

Assuming as before that N follows a normal distribution, we reach $\sigma^2(N)\overline{p}^2 = N\overline{p}^2$. Further, assuming that the individual weights, p_i, follow a Gaussian distribution (which in some cases is far from the truth) we have $N^2\sigma^2(\overline{p}) = \sigma^2(\sum_i p_i) = N\sigma^2(p_i)$ and reach

$$\sigma^2(I) = N\left(\overline{p}^2 + \sigma^2(p_i)\right). \tag{2.6}$$

The statistical variance of the p_i's is estimated by $\sigma^2(p_i) \approx (\sum_i p_i^2 - N\overline{p}^2)/(N-1)$. The resulting variance then reads

$$\sigma^2(I) = \frac{N}{N-1}\left(\sum_i p_i^2 - \overline{p}^2\right). \tag{2.7}$$

For almost any positive value of N, this is very well approximated by the simple expression

$$\sigma^2(I) \approx \sum_i p_i^2. \tag{2.8}$$

As a consistency check, we note that for all p_i equal, this reduces to eq. (2.4)

In order to compute the intensities and uncertainties, the monitor/detector components in McStas will keep track of $N = \sum_i p_i^0$, $I = \sum_i p_i^1$, and $M_2 = \sum_i p_i^2$.

2.3. Weight factor transformations during a Monte Carlo choice

When a Monte Carlo choice must be performed, *e.g.* when the initial energy and direction of the neutron ray is decided at the source, it is important to adjust the neutron weight

[1] This is not correct in a situation where the detector counts a large fraction of the neutron rays in the simulation, but we will neglect that for now.

so that the combined effect of neutron weight change and Monte Carlo probability of making this particular choice equals the actual physical properties we like to model.

Let us follow up on the simple example of transmission. The probability of transmitting the real neutron is P, but we make the Monte Carlo choice of transmitting the neutron ray each time: $f_{\mathrm{MC}} = 1$. This must be reflected on the choice of weight multiplier $\pi_j = P$. Of course, one could simulate without weight factor transformation, in our notation written as $f_{\mathrm{MC}} = P, \pi_j = 1$. To generalize, weight factor transformations are given by the master equation

$$f_{\mathrm{MC}} \pi_j = P. \tag{2.9}$$

This probability rule is general, and holds also if, e.g., it is decided to transmit only half of the rays ($f_{\mathrm{MC}} = 0.5$). An important different example is elastic scattering from a powder sample, where the Monte-Carlo choices are the particular powder line to scatter from, the scattering position within the sample and the final neutron direction within the Debye-Scherrer cone. This weight transformation is much more complex than described above, but still boils down to obeying the master transformation rule 2.9.

2.3.1. Direction focusing

An important application of weight transformation is direction focusing. Assume that the sample scatters the neutron rays in many directions. In general, only neutron rays in some of these directions will stand any chance of being detected. These directions we call the *interesting directions*. The idea in focusing is to avoid wasting computation time on neutrons scattered in the other directions. This trick is an instance of what in Monte Carlo terminology is known as *importance sampling*.

If *e.g.* a sample scatters isotropically over the whole 4π solid angle, and all interesting directions are known to be contained within a certain solid angle interval $\mathbf{\Delta\Omega}$, only these solid angles are used for the Monte Carlo choice of scattering direction. This implies $f_{\mathrm{MC}}(\Delta\Omega) = 1$. However, if the physical events are distributed uniformly over the unit sphere, we would have $P(\Delta\Omega) = \Delta\Omega/(4\pi)$, according to Eq. (2.9). One thus ensures that the mean simulated intensity is unchanged during a "correct" direction focusing, while a too narrow focusing will result in a lower (*i.e.* wrong) intensity, since we cut neutrons rays that should have reached the final detector.

2.4. Adaptive and Stratified sampling

Another strategy to improve sampling in simulations is *adaptive importance sampling* (also called variance reduction technique), where McStas during the simulations will determine the most interesting directions and gradually change the focusing according to that. Implementation of this idea is found in the **Source_adapt** and **Source_Optimizer** components.

An other class of efficiency improvement technique is the so-called *stratified sampling*. It consists in partitioning the event distributions in representative sub-spaces, which are then all sampled individually. The advantage is that we are then sure that each sub-space

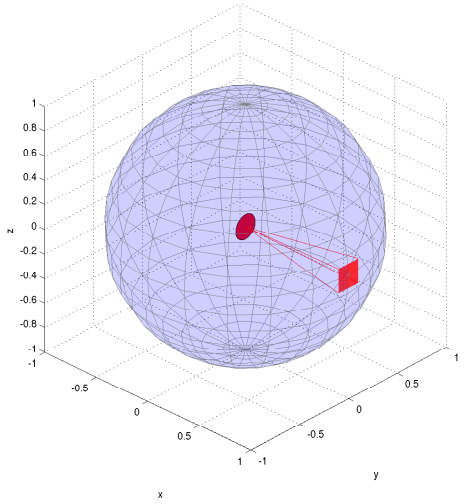


Figure 2.1.: Illustration of the effect of direction focusing in McStas. Weights of neutrons emitted into a certain solid angle are scaled down by the full unit sphere area.

is well represented in the final integrals. This means that instead of shooting N events, we define D partitions and shoot $r = N/D$ events in each partition. In conjunction with adaptive sampling, we may define partitions so that they represent 'interesting' distributions, e.g. from events scattered on a monochromator or a sample. The sum of partitions should equal the total space integrated by the Monte Carlo method, and each partition must be sampled randomly.

In the case of McStas, an ad-hoc implementation of adaptive stratified is used when repeating events, such as in the Virtual sources (Virtual_input, Vitess_input, Virtual_mcnp_input, Virtual_tripoli4_input) and when using the SPLIT keyword in the TRACE section on instrument descriptions. We emphasize here that the number of repetitions r should not exceed the dimensionality of the Monte Carlo integration space (which is $d = 10$ for neutron events) and the dimensionality of the partition spaces, i.e. the number of random generators following the stratified sampling location in the instrument.

2.5. Accuracy of Monte Carlo simulations

When running a Monte Carlo, the meaningful quantities are obtained by integrating random events into a single value (e.g. flux), or onto an histogram grid. The theory [Jam80] shows that the accuracy of these estimates is a function of the space dimension d and the number of events N. For large numbers N, the central limit theorem provides an estimate of the relative error as $1/\sqrt{N}$. However, the exact expression depends on the random distributions.

Records	Accuracy
10^3	10 %
10^4	2.5 %
10^5	1 %
10^6	0.25 %
10^7	0.05 %

Table 2.1.: Accuracy estimate as a function of the number of statistical events used to estimate an integral with McStas.

McStas uses a space with $d = 10$ parameters to describe neutrons (position, velocity, spin, time). We show in Table 2.1 a rough estimate of the accuracy on integrals as a function of the number of records reaching the integration point. This stands both for integrated flux, as well as for histogram bins - for which the number of events per bin should be used for N.

3. Source components

McStas contains a number of different source components, and any simulation will usually contain exactly one of these sources. The main function of a source is to determine a set of initial parameters $(\mathbf{r}, \mathbf{v}, t)$ for each neutron ray. This is done by Monte Carlo choices from suitable distributions. For example, in most present sources the initial position is found from a uniform distribution over the source surface, which can be chosen to be either circular or rectangular. The initial neutron velocity is selected within an interval of either the corresponding energy or the corresponding wavelength. Polarization is not relevant for sources, and we initialize the neutron average spin to zero: $\mathbf{s} = (0, 0, 0)$.

For time-of-flight sources, the choice of the emission time, t, is being made on basis of detailed analytical expressions. For other sources, t is set to zero. In the case one would like to use a steady state source with time-of-flight settings, the emission time of each neutron ray should be determined using a Monte Carlo choice. This may be achieved by the `EXTEND` keyword in the instrument description source as in the example below:

```
1  TRACE
2
3  COMPONENT MySource=Source_gen(...) AT (...)
4  EXTEND
5  %{
6    t = 1e-3*randpm1(); /* set time to +/- 1 ms */
7  %}
```

3.0.1. Neutron flux

The flux of the sources deserves special attention. The total neutron intensity is defined as the sum of weights of all emitted neutron rays during one simulation (the unit of total neutron weight is thus neutrons per second). The flux, ψ, at an instrument is defined as intensity per area perpendicular to the beam direction.

The source flux, Φ, is defined in different units: the number of neutrons emitted per second from a 1 cm^2 area on the source surface, with direction within a 1 ster. solid angle, and with wavelength within a 1 Å interval. The total intensity of real neutrons emitted towards a given diaphragm (units: n/sec) is therefore (for constant Φ):

$$I_{\text{total}} = \Phi A \Delta\Omega \Delta\lambda, \tag{3.1}$$

where A is the source area, $\Delta\Omega$ is the solid angle of the diaphragm as seen from the source surface, and $\Delta\lambda$ is the width of the wavelength interval in which neutrons are emitted (assuming a uniform wavelength spectrum).

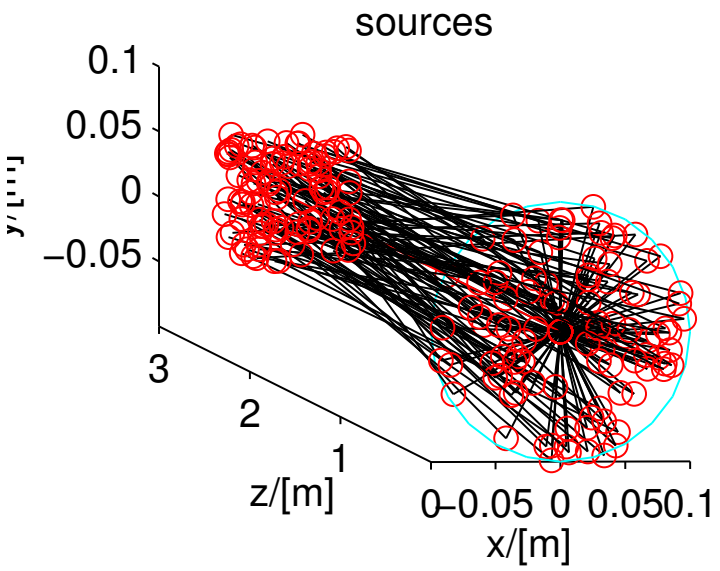


Figure 3.1.: A circular source component (at z=0) emitting neutron events randomly, either from a model, or from a data file.

The simulations are performed so that detector intensities are independent of the number of neutron histories simulated (although more neutron histories will give better statistics). If N_{sim} denotes the number of neutron histories to simulate, the initial neutron weight p_0 must be set to

$$p_0 = \frac{N_{\mathrm{total}}}{N_{\mathrm{sim}}} = \frac{\Phi(\lambda)}{N_{\mathrm{sim}}} A\Omega\Delta\lambda, \tag{3.2}$$

where the source flux is now given a λ-dependence.

As a start, we recommend new McStas users to use the **Source_simple** component. Slightly more realistic sources are **Source_Maxwell_3** for continuous sources or **Moderator** for time-of-flight sources.

Optimizers can dramatically improve the statistics, but may occasionally give wrong results, due to misleaded optimization. You should always check such simulations with (shorter) non-optimized ones.

Other ways to speed-up simulations are to read events from a file. See section 3.10 for details.

3.1. The `Source_simple` McStas Component

A circular neutron source with flat energy spectrum and arbitrary flux

Identification

- **Author:** Kim Lefmann
- **Origin:** Risoe
- **Date:** October 30, 1997

Description

The routine is a circular neutron source, which aims at a square target centered at the beam (in order to improve MC-acceptance rate). The angular divergence is then given by the dimensions of the target. The neutron energy is uniformly distributed between lambda0-dlambda and lambda0+dlambda or between E0-dE and E0+dE. The flux unit is specified in n/cm2/s/st/energy unit (meV or Å).

This component replaces Source_flat, Source_flat_lambda, Source_flux and Source_flux_lambda.

Example: Source_simple(radius=0.1, dist=2, focus_xw=.1, focus_yh=.1, E0=14, dE=2)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	m	Radius of circle in (x,y,0) plane where neutrons are generated.	0.1
yheight	m	Height of rectangle in (x,y,0) plane where neutrons are generated.	0
xwidth	m	Width of rectangle in (x,y,0) plane where neutrons are generated.	0
dist	m	Distance to target along z axis.	0
focus_xw	m	Width of target	.045
focus_yh	m	Height of target	.12
E0	meV	Mean energy of neutrons.	0
dE	meV	Energy half spread of neutrons (flat or gaussian sigma).	0
lambda0	Å	Mean wavelength of neutrons.	0
dlambda	Å	Wavelength half spread of neutrons.	0
flux	1/(s*cm**2*st*energy unit)	Flux per energy unit, Å or meV if flux=0, the source emits 1 in 4*PI whole space.	1

Name	Unit	Description	Default
gauss	1	Gaussian (1) or Flat (0) energy/wavelength distribution	0
target_index	1	relative index of component to focus at, e.g. next is +1 this is used to compute 'dist' automatically.	1

Links

- Component source code found in file `Source_simple.comp`.

A simple continuous source with a flat energy/wavelength spectrum

This component is a simple source with an energy distribution which is uniform in the range $E_0 \pm dE$ (alternatively: a wavelength distribution in the range $\lambda_0 \pm d\lambda$). This component is not used for detailed time-of-flight simulations, so we put $t = 0$ for all neutron rays.

The initial neutron ray position is chosen randomly from within a circle of radius r_s in the $z = 0$ plane. This geometry is a fair approximation of a cylindrical cold/thermal source with the beam going out along the cylinder axis.

The initial neutron ray direction is focused onto a rectangular target of width w, height h, parallel to the xy plane placed at $(0, 0, z_{foc})$.

The initial weight of the created neutron ray, p_0, is set to the energy-integrated flux, Ψ, times the source area, πr_s^2 times a solid-angle factor, which is basically the solid angle of the focusing rectangle. See also the section 3.0.1 on source flux.

This component replaces the obsolete components Source_flux_lambda, Source_flat, Source_flat_lambda, and Source_flux.

3.2. The `Source_div` McStas Component

Neutron source with Gaussian or uniform divergence

Identification

- **Author:** KL
- **Origin:** Risoe
- **Date:** November 20, 1998

Description

The routine is a rectangular neutron source, which has a gaussian or uniform divergent output in the forward direction. The neutron energy is distributed between lambda0-dlambda and lambda0+dlambda or between E0-dE and E0+dE. The flux unit is specified

in n/cm2/s/st/energy unit (meV or Å). In the case of uniform distribution (gauss=0), angles are uniformly distributed between -focus_aw and +focus_aw as well as -focus_ah and +focus_ah. For Gaussian distribution (gauss=1), 'focus_aw' and 'focus_ah' define the FWHM of a Gaussian distribution. Energy/wavelength distribution is also Gaussian.

Example: Source_div(xwidth=0.1, yheight=0.1, focus_aw=2, focus_ah=2, E0=14, dE=2, gauss=0)

%VALIDATION

```
Feb 2005: tested by Kim Lefmann    (o.k.)
```

Apr 2005: energy distribution used in external tests of Fermi choppers (o.k.) Jun 2005: wavelength distribution used in external tests of velocity selectors (o.k.) Validated by: K. Lieutenant

%BUGS distribution is uniform in (hor. and vert.) angle (relative to moderator normal), therefore not suited for large angles

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xwidth	m	Width of source	
yheight	m	Height of source	
focus_aw	deg	FWHM (Gaussian) or maximal (uniform) horz. width divergence	
focus_ah	deg	FWHM (Gaussian) or maximal (uniform) vert. height divergence	
E0	meV	Mean energy of neutrons.	0.0
dE	meV	Energy half spread of neutrons.	0.0
lambda0	Å	Mean wavelength of neutrons (only relevant for E0=0)	0.0
dlambda	Å	Wavelength half spread of neutrons.	0.0
gauss	0|1	Criterion: 0: uniform, 1: Gaussian distributions	0
flux	1/(s cm 2 st energy_unit)	flux per energy unit, Å or meV	1

Links

- Component source code found in file `Source_div.comp`.

A continuous source with specified divergence

Source_div is a rectangular source, $w \times h$, which emits a beam of a specified divergence around the direction of the z axis. The beam intensity is uniform over the whole of the source, and the energy (or wavelength) distribution of the beam is uniform over the specified energy range $E_0 \pm \Delta E$ (in meV), or alternatively the wavelength range $\lambda_0 \pm \delta\lambda$ (in Å).

The source divergences are δ_h and δ_v (FWHM in degrees). If the `gauss` flag is set to 0 (default), the divergence distribution is uniform, otherwise it is Gaussian.

This component may be used as a simple model of the beam profile at the end of a guide or at the sample position.

3.3. The `Source_Maxwell_3` McStas Component

Source with up to three Maxwellian distributions

Identification

- **Author:** Kim Lefmann
- **Origin:** Risoe
- **Date:** March 2001

Description

A parametrised continuous source for modelling a (cubic) source with (up to) 3 Maxwellian distributions. The source produces a continuous spectrum. The sampling of the neutrons is uniform in wavelength.

Units of flux: neutrons/cm^2/second/ster (McStas units are in general neutrons/second)

```
Example:  PSI cold source T1=150.42 K / 2.51 AA     I1 = 3.67 E11
T2=38.74 K / 4.95 AA      I2 = 3.64 E11
T3=14.84 K / 9.5 AA       I3 = 0.95 E11
```

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
size	m	Edge of cube shaped source (for backward compatibility)	0
yheight	m	Height of rectangular source	0
xwidth	m	Width of rectangular source	0
Lmin	Å	Lower edge of lambda distribution	

Name	Unit	Description	Default
Lmax	Å	Upper edge of lambda distribution	
dist	m	Distance from source to focusing rectangle; at (0,0,dist)	
focus_xw	m	Width of focusing rectangle	
focus_yh	m	Height of focusing rectangle	
T1	K	1st temperature of thermal distribution	
T2	K	2nd temperature of thermal distribution	300
T3	K	3nd temperature of - - -	300
I1	1/(cm**2*st)	flux, 1 (in flux units, see above)	
I2	1/(cm**2*st)	flux, 2 (in flux units, see above)	0
I3	1/(cm**2*st)	flux, 3 - - -	0
target_index	1	relative index of component to focus at, e.g. next is +1 this is used to compute 'dist' automatically.	1
lambda0	Å	Mean wavelength of neutrons.	0
dlambda	Å	Wavelength spread of neutrons.	0

Links

- Component source code found in file `Source_Maxwell_3.comp`.

A continuous source with a Maxwellian spectrum

This component is a source with a Maxwellian energy/wavelength distribution sampled in the range λ_{low} to λ_{high}. The initial neutron ray position is chosen randomly from within a rectangle of area $h \times w$ in the $z = 0$ plane. The initial neutron ray direction is focused within a solid angle, defined by a rectangular target of width xw, height yh, parallel to the xy plane placed at $(0, 0, d_{\text{foc}})$. The energy distribution used is a sum of 1, 2, or 3 Maxwellians with temperatures T_1 to T_3 and integrated intensities I_1 to I_3.

For one single Maxwellian, the intensity in a small wavelength interval $[\lambda, \lambda + d\lambda]$ is $I_1 M(\lambda, T_1) d\lambda$ where $M(\lambda, T_1) = 2\alpha^2 \exp(-\alpha/\lambda^2)/\lambda^5$ is the normalized Maxwell distribution ($\alpha = 949.0$ K Å$^2/T_1$). The initial weight of the created neutron ray, p_0, is calculated according to Eq. (3.2), with $\Psi(\lambda)$ replaced by $\sum_{j=1}^{3} I_j M(\lambda, T_j)$.

The component **Source_gen** (see section 3.4) works on the same principle, but provides more options concerning wavelength/energy range specifications, shape, etc.

Some example transmission data files usable alongside sources like this one are given in Table 4.5.

3.4. The `Source_gen` McStas Component

Circular/squared neutron source with flat or Maxwellian energy/wavelength spectrum

Identification

- **Author:** Emmanuel Farhi, Kim Lefmann
- **Origin:** ILL/Risoe
- **Date:** Aug 27, 2001

Description

This routine is a neutron source (rectangular or circular), which aims at a square target centered at the beam (in order to improve MC-acceptance rate). The angular divergence is then given by the dimensions of the target. However, it may be directly set using the 'focus-aw' and 'focus_ah' parameters.

The neutron energy/wavelength is distributed uniformly in wavelength between Emin=E0-dE and Emax=E0+dE or Lmin=lambda0-dlambda and Lmax=lambda0+dlambda. The I1 may be either arbitrary (I1=0), or specified in neutrons per steradian per square cm per Å per s. A Maxwellian spectra may be selected if you give the source temperatures (up to 3).

Finally, a file with the flux as a function of the wavelength [lambda(Å) flux(n/s/cm^2/st/Å)] may be used with the 'flux_file' parameter. Format is 2 columns free text.

Additional distributions for the horizontal and vertical phase spaces distributions (position-divergence) may be specified with the 'xdiv_file' and 'ydiv_file' parameters. Format is free text, requiring a comment line '# xylimits: pos_min pos_max div_min div_max' to set the axis of the distribution matrix. All these files may be generated using standard monitors (better in McStas/PGPLOT format), e.g.: Monitor_nD(options="auto lambda per cm2") Monitor_nD(options="x hdiv, all auto") Monitor_nD(options="y vdiv, all auto")

The source shape is defined by its radius, or can alternatively be squared if you specify non-zero yheight and xwidth parameters. The beam is divergence uniform,. The source may have a thickness, which will broaden the default zero time distribution.

Usage example: Source_gen(radius=0.1,lambda0=2.36,dlambda=0.16,T1=20,I1=1e13,focus_xw=
Source_gen(yheight=0.1,xwidth=0.1,Emin=1,Emax=3,I1=1e13,verbose=1,focus_xw=0.01,focus_y
EXTEND %{ t = rand0max(1e-3); // set time from 0 to 1 ms for TOF instruments. %}

Some neutron facility parameters:

```
PSI cold source     T1=296.2,I1=8.5E11, T2=40.68,I2=5.2E11
ILL VCS cold source T1=216.8,I1=1.24e+13,T2=33.9,I2=1.02e+13
(H1, 58 MW)  T3=16.7 ,I3=3.0423e+12
ILL HCS cold source T1=413.5,I1=10.22e12,T2=145.8,I2=3.44e13
(H5, 58 MW)  T3=40.1 ,I3=2.78e13
ILL Thermal tube    T1=683.7,I1=0.5874e+13,T2=257.7,I2=2.5099e+13
(H12, 58 MW) T3=16.7 ,I3=1.0343e+12
ILL Hot source      T1=1695, I1=1.74e13,T2=708,  I2=3.9e12 (58MW)
HZB cold source     T1=43.7 ,I1=1.4e12, T2=137.2,I2=2.08e12,radius=.155 (10MW)
```

```
HZB bi-spectral     T1=43.7, I1=1.4e12, T2=137.2,I2=2.08e12,T3=293.0,I3=1.77e12
HZB thermal tube    T1=293.0,I1=2.64e12 (10MW)
FRM2 cold,20MW      T1=35.0, I1=9.38e12,T2=547.5,I2=2.23e12,T3=195.4,I3=1.26e13
FRM2 thermal,20MW   T1=285.6,I1=3.06e13,T2=300.0,I2=1.68e12,T3=429.9,I3=6.77e12
LLB cold,14MW       T1=220,  I1=2.09e12,T2=60,   I2=3.83e12,T3=20,   I3=1.04e12
TRIGA thermal 1MW   T1=300,  I1=3.5e11 (scale by thermal power in MW)
```

%VALIDATION Feb 2005: output cross-checked for 3 Maxwellians against VITESS source I(lambda), I(hor_div), I(vert_div) identical in shape and absolute values Validated by: K. Lieutenant

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
flux_file	str	Name of a two columns [lambda flux] text file that contains the wavelength distribution of the flux in either [1/(s*cm**2*st)] or [1/(s*cm**2*st*Å)] (see flux_file_perAA flag) Comments (#) and further columns are ignored. Format is compatible with McStas/PGPLOT wavelength monitor files. When specified, temperature and intensity values are ignored.	"NULL"
xdiv_file	str	Name of the x-horiz. divergence distribution file, given as a free format text matrix, preceeded with a line '# xylimits: xmin xmax xdiv_min xdiv_max'	"NULL"
ydiv_file	str	Name of the y-vert. divergence distribution file, given as a free format text matrix, preceeded with a line '# xylimits: ymin ymax ydiv_min ydiv_max'	"NULL"
radius	m	Radius of circle in (x,y,0) plane where neutrons are generated. You may also use 'yheight' and 'xwidth' for a square source	0.0
dist	m	Distance to target along z axis.	0
focus_xw	m	Width of target.	0.045
focus_yh	m	Height of target.	0.12
focus_aw	deg	maximal (uniform) horz. width divergence	0

Name	Unit	Description	Default
focus_ah	deg	maximal (uniform) vert. height divergence	0
E0	meV	Mean energy of neutrons.	0
dE	meV	Energy spread of neutrons, half width.	0
lambda0	Å	Mean wavelength of neutrons.	0
dlambda	Å	Wavelength spread of neutrons,half width	0
I1	1/(cm**2*sr)	Source flux per solid angle, area and Å if I1=0, the source emits 1 in 4*PI whole space.	1
yheight	m	Source y-height, then does not use radius parameter	0.1
xwidth	m	Source x-width, then does not use radius parameter	0.1
verbose	0/1	display info about the source. -1 inactivate source.	0
T1	K	Temperature of the Maxwellian source, 0=none	0
flux_file_perAA	1	When true (1), indicates that flux file data is already per Aangstroem. If false, file data is per wavelength bin.	0
flux_file_log	1	When true, will transform the flux table in log scale to improve the sampling.	0
Lmin	Å	Minimum wavelength of neutrons	0
Lmax	Å	Maximum wavelength of neutrons	0
Emin	meV	Minimum energy of neutrons	0
Emax	meV	Maximum energy of neutrons	0
T2	K	Second Maxwellian source Temperature, 0=none	0
I2	1/(cm**2*sr)	Second Maxwellian Source flux	0
T3	K	Third Maxwellian source Temperature, 0=none	0
I3	1/(cm**2*sr)	Third Maxwellian Source flux	0
zdepth	m	Source z-zdepth, not anymore flat	0
target_index	1	relative index of component to focus at, e.g. next is +1 this is used to compute 'dist' automatically.	1

Links

- Component source code found in file `Source_gen.comp`.

Source Name	T_1	I_1	T_2	I_2	T_3	I_3	factor
PSI cold source	150.4	3.67e11	38.74	3.64e11	14.84	0.95e11	* I_{target} (mA)
ILL VCS (H1)	216.8	1.24e13	33.9	1.02e13	16.7	3.042e12	58MW
ILL HCS (H5)	413.5	10.22e12	145.8	3.44e13	40.1	2.78e13	58MW
ILL Thermal(H2)	683.7	5.874e12	257.7	2.51e13	16.7	1.034e12	58MW, /2.25
ILL Hot source	1695	1.74e13	708	3.9e12			58MW
PIK cold source	204.6	5.38e12	73.9	7 2.50e12	23.9	9.51e12	
HZB cold source	43.7	1.4e12	137.2	2.08e12			10MW, radius=.155
HZB bi-spectral	43.7	1.4e12	137.2	2.08e12	293	1.77e12	10MW
HZB thermal tube	293.0	2.64e12					10MW
FRM2 cold	35.0	9.38e12	547.5	2.23e12	195.4	1.26e13	20MW
FRM2 thermal	285.6	3.06e13	300.0	1.68e12	429.9	6.77e12	20MW
LLB cold,14MW	220	2.09e12	60	3.83e12	20	1.04e12	14MW
TRIGA thermal	300	3.5e11					1MW

Table 3.5.: Flux parameters for present sources used in components Source_gen and Source_Maxwell_3. For some cases, a correction factor to the intensity should be used to reach measured data; for the PSI cold source, this correction factor is the beam current, I_{target}, which is currently of the order 1.2 mA.

- P. Ageron, Nucl. Inst. Meth. A 284 (1989) 197

A general continuous source

This component is a continuous neutron source (rectangular or circular), which aims at a rectangular target centered at the beam. The angular divergence is given by the dimensions of the target. The shape may be rectangular (dimension h and w), or a disk of radius r. The wavelength/energy range to emit is specified either using center and half width, or using minimum and maximum boundaries, alternatively for energy and wavelength. The flux spectrum is specified with the same Maxwellian parameters as in component Source_Maxwell_3 (refer to section 3.3).

Maxwellian parameters for some continuous sources are given in Table 3.5. As nobody knows exactly the characteristics of the sources (it is not easy to measure spectrum there), these figures should be used with caution.

3.5. The `Moderator` McStas Component

A simple pulsed source for time-of-flight.

Identification

- **Author:** KN, M.Hagen
- **Origin:** Risoe
- **Date:** August 1998

Description

Produces a simple time-of-flight spectrum, with a flat energy distribution

```
Example: Moderator(radius = 0.0707, dist = 9.035, focus_xw = 0.021, focus_yh = 0.021,
```

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	m	Radius of source	0.07
Emin	meV	Lower edge of energy distribution	
Emax	meV	Upper edge of energy distribution	
dist	m	Distance from source to the focusing rectangle	0
focus_xw	m	Width of focusing rectangle	0.02
focus_yh	m	Height of focusing rectangle	0.02
t0	mus	decay constant for low-energy neutrons	37.15
Ec	meV	Critical energy, below which the flux decay is constant	9.0
gamma	meV	energy dependence of decay time	39.1
target_index	1	relative index of component to focus at, e.g. next is +1 this is used to compute 'dist' automatically.	1
flux	1/(s cm 2 st meV)	flux	1

Links

- Component source code found in file `Moderator.comp`.

A time-of-flight source (pulsed)

Name:	Moderator
Author:	(System) Mark Hagen, SNS
Input parameters	r_s, E_0, E_1, z_f, w, h, τ_0, E_c, γ
Optional parameters	
Notes	

The simple time-of-flight source component **Moderator** resembles the source component **Source_simple** described in 3.1. **Moderator** is circular with radius r_s and focuses on a rectangular target of area $w \times h$ in a distance z_f. The initial velocity is chosen with a linear distribution within an interval, defined by the minimum and maximum energies, E_0 and E_1, respectively.

The initial time of the neutron is determined on basis of a simple heuristical model for the time dependence of the neutron intensity from a time-of-flight source. For all neutron energies, the flux decay is assumed to be exponential,

$$\Psi(E,t) = \exp(-t/\tau(E)), \tag{3.3}$$

where the decay constant is given by

$$\tau(E) = \begin{cases} \tau_0 & ; E < E_c \\ \tau_0/[1+(E-E_c)^2/\gamma^2] & ; E \geq E_c \end{cases} \tag{3.4}$$

The decay parameters are τ_0 (in μs), E_c, and γ (both in meV).

Other pulsed source models are available from contributed components. See section 3.10.

3.6. The `Source_adapt` McStas Component

Neutron source with adaptive importance sampling

Identification

- **Author:** Kristian Nielsen
- **Origin:** Risoe
- **Date:** 1999

Description

Rectangular source with flat energy or wavelength distribution that uses adaptive importance sampling to improve simulation efficiency. Works together with the Adapt_check component.

The source divides the three-dimensional phase space of (energy, horizontal position, horizontal divergence) into a number of rectangular bins. The probability for selecting neutrons from each bin is adjusted so that neutrons that reach the Adapt_check component with high weights are emitted more frequently than those with low weights. The adjustment is made so as to attemt to make the weights at the Adapt_check components equal.

Focusing is achieved by only emitting neutrons towards a rectangle perpendicular to and placed at a certain distance along the Z axis. Focusing is only approximate (for simplicity); neutrons are also emitted to pass slightly above and below the focusing rectangle, more so for wider focusing.

In order to prevent false learning, a parameter beta sets a fraction of the neutrons that are emitted uniformly, without regard to the adaptive distribution. The parameter alpha sets an initial fraction of neutrons that are emitted with low weights; this is done to prevent early neutrons with rare initial parameters but high weight to ruin the statistics before the component adapts its distribution to the problem at hand. Good general-purpose values for these parameters are alpha = beta = 0.25.

%VALIDATION This component is not validated. It does not work properly with MPI.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
N_E	1	Number of bins in energy (or wavelength) dimension	20
N_xpos	1	Number of bins in horizontal position	20
N_xdiv	1	Number of bins in horizontal divergence	20

Name	Unit	Description	Default
xmin	m	Left edge of rectangular source	0
xmax	m	Right edge	0
ymin	m	Lower edge	0
ymax	m	Upper edge	0
xwidth	m	Width of source	0
yheight	m	Height of source	0
filename	string	Optional filename for adaptive distribution output	0
dist	m	Distance to target rectangle along z axis	0
focus_xw	m	Width of target	0.05
focus_yh	m	Height of target	0.1
E0	meV	Mean energy of neutrons	0
dE	meV	Energy spread (energy range is from E0-dE to E0+dE)	0
lambda0	Å	Mean wavelength of neutrons (if energy not specified)	0
dlambda	Å	Wavelength spread half width	0
flux	1/(cm^2*Å*st)	Absolute source flux	1e13
target_index	1	relative index of component to focus at, e.g. next is +1 this is used to compute 'dist' automatically.	1
alpha	1	Learning cut-off factor (0 < alpha <= 1)	0.25
beta	1	Aggressiveness of adaptive algorithm (0 < beta <= 1)	0.25

Links

- Component source code found in file `Source_adapt.comp`.

A neutron source with adaptive importance sampling

Source_adapt is a neutron source that uses adaptive importance sampling to improve the efficiency of the simulations. It works by changing on-the-fly the probability distributions from which the initial neutron state is sampled so that samples in regions that contribute much to the accuracy of the overall result are preferred over samples that contribute little. The method can achieve improvements of a factor of ten or sometimes several hundred in simulations where only a small part of the initial phase space contains useful neutrons. This component uses the correlation between neutron energy, initial direction and initial position.

The physical characteristics of the source are similar to those of **Source_simple** (see section 3.1). The source is a thin rectangle in the x-y plane with a flat energy spectrum in a user-specified range. The flux, Φ, per area per steradian per Ångström per second

is specified by the user.

The initial neutron weight is given by Eq. (3.2) using $\Delta\lambda$ as the total wavelength range of the source. A later version of this component will probably include a λ-dependence of the flux.

We use the input parameters *dist*, *xw*, and *yh* to set the focusing as for Source_simple (section 3.1). The energy range will be from $E_0 - dE$ to $E_0 + dE$. *filename* is used to give the name of a file in which to output the final sampling destribution, see below. $N_{\rm eng}$, $N_{\rm pos}$, and $N_{\rm div}$ are used to set the number of bins in each dimensions. Good general-purpose values for the optimization parameters are $\alpha = \beta = 0.25$. The number of bins to choose will depend on the application. More bins will allow better adaption of the sampling, but will require more neutron histories to be simulated before a good adaption is obtained. The output of the sampling distribution is only meant for debugging, and the units on the axis are not necessarily meaningful. Setting the filename to `NULL` disables the output of the sampling distribution.

Optimization disclaimer

A warning is in place here regarding potentially wrong results using optimization techniques. It is highly recommended in any case to benchmark 'optimized' simulations against non-optimized ones, checking that obtained results are the same, but hopefully with a much improved statistics.

The adaption algorithm

The adaptive importance sampling works by subdividing the initial neutron phase space into a number of equal-sized bins. The division is done on the three dimensions of energy, horizontal position, and horizontal divergence, using $N_{\rm eng}$, $N_{\rm pos}$, and $N_{\rm div}$ number of bins in each dimension, respectively. The total number of bins is therefore

$$N_{\rm bin} = N_{\rm eng} N_{\rm pos} N_{\rm div} \tag{3.5}$$

Each bin i is assigned a sampling weight w_i; the probability of emitting a neutron within bin i is

$$P(i) = \frac{w_i}{\sum_{j=1}^{N_{\rm bin}} w_j} \tag{3.6}$$

In order to avoid false learning, the sampling weight of a bin is kept larger than $w_{\rm min}$, defined as

$$w_{\rm min} = \frac{\beta}{N_{\rm bin}} \sum_{j=1}^{N_{\rm bin}} w_j, \qquad 0 \le \beta \le 1 \tag{3.7}$$

This way a (small) fraction β of the neutrons are sampled uniformly from all bins, while the fraction $(1-\beta)$ are sampled in an adaptive way.

Compared to a uniform sampling of the phase space (where the probability of each bin is $1/N_{\rm bin}$), the neutron weight must be adjusted as given by (2.9)

$$\pi_1 = \frac{P_1}{f_{\rm MC,1}} = \frac{1/N_{\rm bin}}{P(i)} = \frac{\sum_{j=1}^{N_{\rm bin}} w_j}{N_{\rm bin} w_i}, \tag{3.8}$$

where P_1 is understood by the "natural" uniform sampling.

In order to set the criteria for adaption, the **Adapt_check** component is used (see section 3.7). The source attemps to sample only from bins from which neutrons are not absorbed prior to the position in the instrument at which **Adapt_check** is placed. Among those bins, the algorithm attemps to minimize the variance of the neutron weights at the **Adapt_check** position. Thus bins that would give high weights at the **Adapt_check** position are sampled more often (lowering the weights), while those with low weights are sampled less often.

Let $\pi = p_{\mathrm{ac}}/p_0$ denote the ratio between the neutron weight p_1 at the **Adapt_check** position and the initial weight p_0 just after the source. For each bin, the component keeps track of the sum Σ of π's as well as of the total number of neutrons n_i from that bin. The average weight at the **Adapt_source** position of bin i is thus Σ_i/n_i.

We now distribute a total sampling weight of β uniformly among all the bins, and a total weight of $(1-\beta)$ among bins in proportion to their average weight Σ_i/n_i at the **Adapt_source** position:

$$w_i = \frac{\beta}{N_{\mathrm{bin}}} + (1-\beta)\frac{\Sigma_i/n_i}{\sum_{j=1}^{N_{\mathrm{bins}}} \Sigma_j/n_j} \tag{3.9}$$

After each neutron event originating from bin i, the sampling weight w_i is updated.

This basic idea can be improved with a small modification. The problem is that until the source has had the time to learn the right sampling weights, neutrons may be emitted with high neutron weights (but low probability). These low probability neutrons may account for a large fraction of the total intensity in detectors, causing large variances in the result. To avoid this, the component emits early neutrons with a lower weight, and later neutrons with a higher weight to compensate. This way the neutrons that are emitted with the best adaption contribute the most to the result.

The factor with which the neutron weights are adjusted is given by a logistic curve

$$F(j) = C\frac{y_0}{y_0 + (1-y_0)e^{-r_0 j}} \tag{3.10}$$

where j is the index of the particular neutron history, $1 \le j \le N_{\mathrm{hist}}$. The constants y_0, r_0, and C are given by

$$y_0 = \frac{2}{N_{\mathrm{bin}}} \tag{3.11}$$

$$r_0 = \frac{1}{\alpha}\frac{1}{N_{\mathrm{hist}}}\log\left(\frac{1-y_0}{y_0}\right) \tag{3.12}$$

$$C = 1 + \log\left(y_0 + \frac{1-y_0}{N_{\mathrm{hist}}}e^{-r_0 N_{\mathrm{hist}}}\right) \tag{3.13}$$

The number α is given by the user and specifies (as a fraction between zero and one) the point at which the adaption is considered good. The initial fraction α of neutron histories are emitted with low weight; the rest are emitted with high weight:

$$p_0(j) = \frac{\Phi}{N_{\mathrm{sim}}} A\Omega\Delta\lambda \frac{\sum_{j=1}^{N_{\mathrm{bin}}} w_j}{N_{\mathrm{bin}} w_i} F(j) \tag{3.14}$$

The choice of the constants y_0, r_0, and C ensure that

$$\int_{t=0}^{N_{\text{hist}}} F(j) = 1 \tag{3.15}$$

so that the total intensity over the whole simulation will be correct

Similarly, the adjustment of sampling weights is modified so that the actual formula used is

$$w_i(j) = \frac{\beta}{N_{\text{bin}}} + (1-\beta)\frac{y_0}{y_0 + (1-y_0)e^{-r_0 j}} \frac{\psi_i/n_i}{\sum_{j=1}^{N_{\text{bins}}} \psi_j/n_j} \tag{3.16}$$

The implementation

The heart of the algorithm is a discrete distribution p. The distribution has N *bins*, $1 \ldots N$. Each bin has a value v_i; the probability of bin i is then $v_i/(\sum_{j=1}^{N} v_j)$.

Two basic operations are possible on the distribution. An *update* adds a number a to a bin, setting $v_i^{\text{new}} = v_i^{\text{old}} + a$. A *search* finds, for given input b, the minimum i such that

$$b \leq \sum_{j=1}^{i} v_j. \tag{3.17}$$

The search operation is used to sample from the distribution p. If r is a uniformly distributed random number on the interval $[0; \sum_{j=1}^{N} v_j]$ then $i = \text{search}(r)$ is a random number distributed according to p. This is seen from the inequality

$$\sum_{j=1}^{i-1} v_j < r \leq \sum_{j=1}^{i} v_j, \tag{3.18}$$

from which $r \in [\sum_{j=1}^{i-1} v_j; v_i + \sum_{j=1}^{i-1} v_j]$ which is an interval of length v_i. Hence the probability of i is $v_i/(\sum_{j=1}^{N} v_j)$. The update operation is used to adapt the distribution to the problem at hand during a simulation. Both the update and the add operation can be performed very efficiently.

As an alternative, you may use the **Source_Optimizer** component (see section 3.8).

3.7. The `Adapt_check` McStas Component

Optimization specifier for the Source_adapt component.

Identification

- **Author:** Kristian Nielsen
- **Origin:** Risoe
- **Date:** 1999

Description

This components works together with the Source_adapt component, and is used to define the criteria for selecting which neutrons are considered "good" in the adaptive algorithm. The name of the associated Source_adapt component in the instrument definition is given as parameter. The component is special in that its position does not matter; all neutrons that have not been absorbed prior to the component are considered "good".

Example: Adapt_check(source_comp="MySource")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
source_comp	string	The name of the Source_adapt component in the instrument definition.	

Links

- Component source code found in file `Adapt_check.comp`.

The adaptive importance sampling monitor

The component **Adapt_check** is used together with the Source_adapt component - see section 3.6 for details. When placed somewhere in an instrument using Source_adapt as a source, the source will optimize for neutrons that reach that point without being absorbed (regardless of neutron position, divergence, wavelength, *etc*).

The Adapt_check component takes as single input parameter *source_comp* the name of the Source_adapt component instance, for example:

```
...
COMPONENT mysource = Source_adapt(...)
...
COMPONENT mycheck = Adapt_check(source_comp = mysource)
...
```

Only one instance of Adapt_check is allowed in an instrument.

We suggest, as alternative method, to make use of the `SPLIT` keyword, as described in the McStas User Manual.

3.8. The `Source_Optimizer` McStas Component

A component that optimizes the neutron flux passing through the Source_Optimizer in order to have the maximum flux at the Monitor_Optimizer position.

Identification

- **Author:** Emmanuel Farhi
- **Origin:** ILL (France)
- **Date:** 17 Sept 1999

Description

Principle: The optimizer first (step 1) computes neutron state parameter limits passing in the Source_Optimizer, and then (step 2) records a Reference source as well as the state (at Source_Optimizer position) of neutrons reaching Monitor. The optimized source is defined as a fraction of the Reference source plus the distribution of 'good' neutrons reaching the Monitor. The optimization then starts (step 3), and focuses new neutrons on the Monitor_Optimizer. In fact it changes 'bad' neutrons into 'good' ones (that reach the Monitor), acting on their position, spin and divergence or velocity. The overall Monitor flux is kept during process. The energy and polarisation distributions are kept during optimization as far as possible during optimisation. The optimization method considers that all neutron parameters - (x,y), (vx,vy,vz) or (vx/v2,vy/v2,v2), (sx,sy,sz) or (sx/s2,sy/s2,s2) - are independent.

Options: The optimized source can be computed regularly ('continuous' option) or only once ('not continuous'). The time spent in steps 1 and 2 can be reduced for a shorter optimization ('auto'). The neutrons passing during steps 1 and 2 can be smoothed for a better neutron weight distribution ('smooth' option).

Source_optimizer can be placed at any position where you want to act on the flux, for instance just after the source. Monitor_Optimizer should be placed at position(s) to optimize. I prefer to put one just before the sample.

Default parameters bins, step, and keep are 10, 10% and 10% respectively. The option string can be empty (""), which stands for default configuration that works fine in usual cases:

options="continuous optimization, auto mode, smooth, SetXY+SetDivV+SetDivS"

Possible options are

```
continuous      for continuous source optimization (default).
verbose         displays optimization process (debug purpose).
auto            uses the shortest possible 'step 1' and 'step 2' and sets 'step' valu
smooth          remove possible spikes generated in steps 1 and 2 (default is smooth)
inactivate      to inactivate the Optimizer.
no or not       revert next option
```

bins=[value=10] set the Number of cells for sampling neutron states step=[value=10] Optimizer step in % of simulation. keep=[value=10] Percentage of initial source distribution that is kept

```
file=[name]     Filename where to save optimized source distributions (no file is generated if
SetXY           Keywords to indicate what may be changed during
SetV            optimisation. Default is position, divergence and spin
SetS            direction ("SetXY+SetDivV+SetdivS"). Choosing the speed
SetDivV         or spin optimization (SetV or SetS) may modify the energy
SetDivS         or polarisation distribution (norm of V and S) as the three components are then
```

Parameters bins, step and keep can also be entered as optional parameters.

EXAMPLE: I use the following settings

optim_s = Source_Optimizer(options="please be clever") (same as empty) (...) Monitor_Optimizer(xmin=-0.05, xmax=0.05, ymin=-0.05, ymax=0.05, optim_comp = "optim_s")

A good optimization needs to record enough non optimized neutrons on Monitor during step 2. Typical enhancement in computation speed is by a factor 20. This component usually works well.

NOTE: You must be aware that in some cases (SetV and SetS), the optimization might sligtly afect the energy or spin distribution of the source. The optimizer tries to do its best anyway. Also, some 'spikes' may sometime appear in monitor signals in the course of the optimization, coming from non-optimized neutrons with original weight. The 'smooth' option minimises this effect (on by default).

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
bins	1	Number of cells for sampling neutron states.	10
step	0-100	Optimizer step in percent of simulation.	0.1
keep	0-100	Percentage of initial source distribution that is kept.	0.1
options	str	string of options. See <b>Description<b>	0

Links

- Component source code found in file `Source_Optimizer.comp`.

A general Optimizer for McStas

The component **Source_Optimizer** is not exactly a source, but rather a neutron beam modifier. It should be positioned after the source, anywhere in the instrument description.

The component optimizes the whole neutron flux in order to achieve better statistics at each **Monitor_Optimizer** location(s) (see section 3.9 for this latter component). It acts on any incoming neutron beam (from any source type), and more than one optimization criteria location can be placed along the instrument.

The usage of the optimizer is very simple, and usually does not require any configuration parameter. Anyway the user can still customize the optimization through various *options*.

In contrast to **Source_adapt**, this optimizer does not record correlations between neutron parameters. Nevertheless it is rather efficient, enabling the user to increase the number of events at optimization criteria locations by typically a factor of 20. Hence, the signal error bars will decrease by a factor 4.5, since the overall flux remains unchanged.

The optimization algorithm

When a neutron reaches the **Monitor_Optimizer** location(s), the component records its previous position (x, y) and speed (v_x, v_y, v_z) when it passed in the **Source_Optimizer**. Some distribution tables of *good* neutrons characteristics are then built.

When a *bad* neutron comes to the **Source_Optimizer** (it would then have few chances to reach **Monitor_Optimizer**), it is changed into a better one. That means that its position and velocity coordinates are translated to better values according to the *good* neutrons distribution tables. The neutron energy ($\sqrt{v_x^2 + v_y^2 + v_z^2}$) is kept (as far as possible).

The **Source_Optimizer** works as follow:

1. First of all, the **Source_Optimizer** determines some limits (*min* and *max*) for variables x, y, v_x, v_y, v_z.

2. Then the component records the non-optimized flux distributions in arrays with *bins* cells (default is 10 cells). This constitutes the *Reference* source.

3. The **Monitor_Optimizer** records the *good* neutrons (that reach it) and communicate an *Optimized* beam requirement to the **Source_Optimizer**. However, retains ' *keep*' percent of the original *Reference* source is sent unmodified (default is 10 %). The *Optimized* source is thus:

   $$\begin{aligned} \textit{Optimized} &= \textit{keep} * \textit{Reference} \\ &+ (1 - \textit{keep}) \text{ [Neutrons that will reach monitor].} \end{aligned}$$

4. The **Source_Optimizer** transforms the *bad* neutrons into *good* ones from the *Optimized* source. The resulting optimised flux is normalised to the non-optimized one:

   $$p_{optimized} = p_{initial} \frac{\text{Reference}}{\text{Optimized}}, \tag{3.19}$$

   and thus the overall flux at **Monitor_Optimizer** location is the same as without the optimizer. Usually, the process sends more *good* neutrons from the *Optimized*

source than that in the *Reference* one. The energy (and velocity) spectra of neutron beam is also kept, as far as possible. For instance, an optimization of v_z will induce a modification of v_x or v_y to try to keep $|\mathbf{v}|$ constant.

5. When the *continuous* optimization option is activated (by default), the process loops to Step (3) every '*step*' percent of the simulation. This parameter is computed automatically (usually around 10 %) in *auto* mode, but can also be set by user.

During steps (1) and (2), some non-optimized neutrons with original weight $p_{initial}$ may lead to spikes on detector signals. This is greatly improved by lowering the weight p during these steps, with the *smooth* option. The component optimizes the neutron parameters on the basis of independant variables (1D phase-space optimization). However, it usually does work fine when these variables are correlated (which is often the case in the course of the instrument simulation). The memory requirements of the component are very low, as no big n-dimensional array is needed.

Using the Source_Optimizer

To use this component, just install the **Source_Optimizer** after a source (but any location is possible afterwards in principle), and use the **Monitor_Optimizer** at a location where you want to reach better statistics.

```
/* where to act on neutron beam */
COMPONENT optim_s = Source_Optimizer(options="")
...
/* where to have better statistics */
COMPONENT optim_m = Monitor_Optimizer(
xmin = -0.05, xmax = 0.05,
ymin = -0.05, ymax = 0.05,
optim_comp = optim_s)
...
/* using more than one Monitor_Optimizer is possible */
```

The input parameter for **Source_Optimizer** is a single *options* string that can contain some specific optimizer configuration settings in clear language. The formatting of the *options* parameter is free, as long as it contains some specific keywords, that can be sometimes followed by values.

The default configuration (equivalent to *options* = "") is

options = "*continuous* optimization, *auto* setting, *keep* = 0.1, *bins* = 0.1, *smooth* spikes, SetXY+SetDivV+SetDivS".

Parameters keep and step should be between 0 and 1. Additionally, you may restrict the optimization to only some of the neutron parameters, using the *SetXY, SetV, SetS, SetDivV, SetDivS* keywords. The keyword modifiers *no* or *not* revert the next option. Other options not shown here are:

```
verbose          displays optimization process (debug purpose).
inactivate       to inactivate the Optimizer.
file=[name]      Filename where to save optimized source distributions
```

The *file* option will save the source distributions at the end of the optimization. If no name is given the component name will be used, and a '.src' extension will be added. By default, no file is generated. The file format is in a McStas 2D record style.

As an alternative, you may use the Source_adapt component (see section 3.6) which performs a 3D phase-space optimization.

3.9. The `Monitor_Optimizer` McStas Component

To be used after the Source_Optimizer component

Identification

- **Author:** Emmanuel Farhi
- **Origin:** ILL (France)
- **Date:** 17 Sept 1999

Description

A component that optimizes the neutron flux passing through the **Source_Optimizer** in order to have the maximum flux at the Monitor_Optimizer position(s). **Source_optimizer** should be placed just after the source. Monitor_Optimizer should be placed at the position to optimize. I prefer to put one just before the sample.

See Source_Optimizer for usage example and additional informations.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
optim_comp	str	name of the Source_Optimizer component in the instrument definition. Do not use quotes (no quotes)	
xmin	m	Lower x bound of monitor opening	-0.1
xmax	m	Upper x bound of monitor opening	0.1
ymin	m	Lower y bound of monitor opening	-0.1
ymax	m	Upper y bound of monitor opening	0.1
xwidth	m	Width of monitor. Overrides xmin,xmax.	0
yheight	m	Height of monitor. Overrides ymin,ymax.	0

Links

- Component source code found in file `Monitor_Optimizer.comp`.
- Source_Optimizer

Optimization locations for the Source_Optimizer

The **Monitor_Optimizer** component works with the **Source_Optimizer** component. See section 3.8 for usage.

The input parameters for **Monitor_Optimizer** are the rectangular shaped opening coordinates x_{min}, x_{max}, y_{min}, y_{max}, and the name of the associated instance of **Source_Optimizer** used in the instrument description file (one word, without quotes).

As many Monitor_Optimizer instances as required may be used in an instrument, for possibly more than one optimization location. Multiple instances may all have an effect on the total intensity.

3.10. Other sources components: contributed pulsed sources, virtual sources (event files)

There are many other source definitions in McStas.

Detailed pulsed source components are available for new facilities in a number of contributed components:

- SNS (**contrib/SNS_source**),
- ISIS (**contrib/ISIS_moderator**),
- ESS-project (**sources/ESS_butterfly**), the current, detailed time-of-flight moderator/target-station model for the European Spallation Source, superseding the earlier **ESS_moderator_sh** and **ESS_moderator** (now in **obsolete**).

When no analytical model (e.g. a Maxwellian distribution) exists, one may have access to measurements, estimated flux distributions, or neutron event files. As of McStas 3.x, the recommended, portable way to read and write such neutron event lists is the MCPL format (see the **misc/MCPL_input** and **misc/MCPL_output** components, section 11.1), which has backends for McStas/McXtrace as well as several other Monte Carlo transport codes (MCNP, PHITS, Geant4, Vitess, SIMRES). The older, code-specific event-file components (**Virtual_input**/**Virtual_output**, **Virtual_mcnp_input**/**Virtual_mcnp_output**, **Virtual_tripoli4_input**/**Virtual_tripoli4_ou** **Vitess_input**/**Vitess_output**) still exist for backward compatibility but have moved to the **obsolete** component category (see the Component Manual's Obsolete chapter); new instrument work should prefer MCPL.

Finally, **optics/Filter_gen** reads a 1D distribution from a file, and may either modify or set the flux according to it. See its entry in the Optics chapter of the Component Manual.

4. Beam optical components: Arms, slits, collimators, and filters

This chapter contains a number of optical components that is used to modify the neutron beam in various ways, as well as the "generic" component **Arm**.

4.1. The `Arm` McStas Component

Arm/optical bench

Identification

- **Author:** Kim Lefmann and Kristian Nielsen
- **Origin:** Risoe
- **Date:** September 1997

Description

An arm does not actually do anything, it is just there to set up a new coordinate system.
Example: Arm()

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default

Links

- Component source code found in file `Arm.comp`.

The generic component

Name:	Arm
Author:	System
Input parameters	(none)
Optional parameters	(none)
Notes	

The component **Arm** is empty; is resembles an optical bench and has no effect on the neutron ray. The purpose of this component is only to provide a standard means of defining a local coordinate system within the instrument definition. Other components may then be positioned relative to the **Arm** component using the McStas meta-language. The use of Arm components in the instrument definitions is not required but is recommended for clarity. **Arm** has no input parameters.

The first Arm instance in an instrument definition may be changed into a `Progress_bar` component in order to display on the fly the simulation progress, and possibly save intermediate results.

4.2. The `Slit` McStas Component

Rectangular/circular slit

Identification

- **Author:** Kim Lefmann and Henrik M. Roennow
- **Origin:** Risoe
- **Date:** June 16, 1997

Description

A simple rectangular or circular slit. You may either specify the radius (circular shape), or the rectangular bounds. No transmission around the slit is allowed.

Example: Slit(xmin=-0.01, xmax=0.01, ymin=-0.01, ymax=0.01) Slit(xwidth=0.02, yheight=0.02) Slit(radius=0.01)

The Slit will issue a warning if run as "closed"

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xmin	m	Lower x bound	UNSET
xmax	m	Upper x bound	UNSET
ymin	m	Lower y bound	UNSET
ymax	m	Upper y bound	UNSET
radius	m	Radius of slit in the z=0 plane, centered at Origin	UNSET
xwidth	m	Width of slit. Overrides xmin,xmax if they are unset.	UNSET

Name	Unit	Description	Default
yheight	m	Height of slit. Overrides ymin,ymax if they are unset.	UNSET

Links

- Component source code found in file `Slit.comp`.

A beam defining diaphragm

The component **Slit** is a very simple construction. It sets up an opening at $z = 0$, and propagates the neutrons onto this plane (by the kernel call PROP_Z0). Neutrons within the slit opening are unaffected, while all other neutrons are discarded by the kernel call ABSORB.

By using Slit, some neutrons contributing to the background in a real experiment will be neglected. These are the ones that scatter off the inner side of the slit, penetrates the slit material, or clear the outer edges of the slit.

The input parameters of **Slit** are the four coordinates, $(x_{\min}, x_{\max}, y_{\min}, y_{\max})$ defining the opening of the rectangle, or the radius r of a circular opening, depending on which parameters are specified.

The slit component can also be used to discard insignificant (*i.e.* very low weight) neutron rays, that in some simulations may be very abundant and therefore time consuming. If the optional parameter p_{cut} is set, all neutron rays with $p < p_{\text{cut}}$ are ABSORB'ed. This use is recommended in connection with **Virtual_output**.

4.3. The `Beamstop` McStas Component

Rectangular/circular beam stop.

Identification

- **Author:** Kristian Nielsen
- **Origin:** Risoe
- **Date:** January 2000

Description

A simple rectangular or circular beam stop. Infinitely thin and infinitely absorbing. The beam stop is by default rectangular. You may either specify the radius (circular shape), or the rectangular bounds.

Example: Beamstop(xmin=-0.05, xmax=0.05, ymin=-0.05, ymax=0.05) Beamstop(radius=0.1)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xmin	m	Lower x bound	-0.05
xmax	m	Upper x bound	0.05
ymin	m	Lower y bound	-0.05
ymax	m	Upper y bound	0.05
xwidth	m	Width of beamstop (x). Overrides xmin, xmax.	0
yheight	m	Height of beamstop (y). Overrides ymin, ymax.	0
radius	m	radius of the beam stop in the z=0 plane, centered at Origo	0

Links

- Component source code found in file `Beamstop.comp`.

A neutron absorbing area

The component **Beamstop** can be seen as the reverse of the **Slit** component. It sets up an area at the $z = 0$ plane, and propagates the neutrons onto this plane (by the kernel call PROP_Z0). Neutrons within this area are ABSORB'ed, while all other neutrons are unaffected.

By using this component, some neutrons contributing to the background in a real experiment will be neglected. These are the ones that scatter off the side of the beamstop, or penetrates the absorbing material. Further, the holder of the beamstop is not simulated.

Beamstop can be either circular or rectangular. The input parameters of **Beamstop** are the four coordinates, $(x_{\min}, x_{\max}, y_{\min}, y_{\max})$ defining the opening of a rectangle, or the radius r of a circle, depending on which parameters are specified.

If the "direct beam" (e.g. after a monochromator or sample) should not be simulated, it is possible to emulate an ideal beamstop so that only the scattered beam is left; without the use of **Beamstop**: This method is useful for instance in the case where only neutrons scattered from a sample are of interest. The example below removes the direct beam and any background signal from other parts of the instrument

```
1 COMPONENT MySample=V_sample(...) AT (...)
2 EXTEND
3 %{
4   if (!SCATTERED) ABSORB;
5 %}
```

4.4. The `Filter_gen` McStas Component

Release: McStas 1.6

This components may either set the flux or change it (filter-like), using an external data filename.

Identification

- **Author:** E. Farhi
- **Origin:** ILL
- **Date:** Dec, 15th, 2002

Description

This component changes the neutron flux (weight) in order to match a reference table in a filename. Typically you may set the neutron flux (source-like), or multiply it using a transmission table (filter-like). The component may be placed after a source, in order to e.g. simulate a real source from a reference table, or used as a filter (BeO) or as a window (Al). The behaviour of the component is specified using the 'options' parameter, or from the filename itself (see below) If the thickness for the transmission data filename D was t0, and a different thickness t1 would be required, then the resulting transmission is:

```
D^(t1/t0).
```

You may use the 'thickness' and 'scaling' parameter for that purpose.

File format: This filename may be of any 2-columns free format (k[Å-1],p), (omega[meV],p) and (lambda[Å],p) where p is the weight. The type of the filename may be written explicitely in the filename, as a comment, or using the 'options' parameter. Non mumerical content in filename is treated as comment (e.g. lines starting with '#' character). A table rebinning and linear interpolation are performed.

EXAMPLE : in order to simulate a PG filter, using the lib/data/HOPG.trm file Filter_gen(xwidth=.1 yheight=.1, filename="HOPG.trm") A Sapphire filter, using the lib/data/Al2O3_sapphire.trm file Filter_gen(xwidth=.1 yheight=.1, filename="Al2O3_sapphire.trm") A Berylium filter, using the lib/data/Be.trm file Filter_gen(xwidth=.1 yheight=.1, filename="Be.trm") an other possibility to simulate a Be filter is to use the PowderN component: PowderN(xwidth=.1, yheight=.1, zdepth=.1, reflections="Be.laz", p_inc=1e-4)

in this filename, the comment line # wavevector multiply sets the behaviour of the component. One may as well have used options="wavevector multiply" in the component instance parameters.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename	str	name of the filename to look at (first two columns data). Data D should rather be sorted (ascending order) and monotonic filename may contain options (see below) as comment	0
options	str	string that can contain: "[k p]" or "wavevector" for filename type, "[omega p]" or "energy", "[lambda p]" or "wavelength", "set" to set the weight according to the table,"multiply" to multiply (instead of set) the weight by factor,"add" to add to current flux,"verbose" to display additional informations.	0
xmin	m	dimension of filter	-0.05
xmax	m	dimension of filter	0.05
ymin	m	dimension of filter	-0.05
ymax	m	dimension of filter	0.05
xwidth	m	Width/diameter of filter). Overrides xmin,xmax.	0
yheight	m	Height of filter. Overrides ymin,ymax.	0
thickness	1	relative thickness. D = D^(thickness).	1
scaling	1	scaling factor. D = D*scaling.	1
verbose	1	Flag to select verbose output.	0

Links

- Component source code found in file `Filter_gen.comp`.
- HOPG.trm filename as an example.

Example transmission data files

Some example transmission data files are provided with McStas in the `MCSTAS/data` directory as `*.trm` files, for use with the `filename` parameter above:

4.5. The `NCrystal_filter` McStas Component

Filter or window (box or cylinder) of a material from NCrystal, which only attenuates the beam. Simple, fast and GPU friendly.

File name	Description
Be.trm	Berylium filter for cold neutron spectrometers (e.g. $k < 1.55$ Å^{-1})
HOPG.trm	Highly oriented pyrolithic graphite for $\lambda/2$ filtering (e.g. thermal beam at $k = 1.64$, 2.662, and 4.1 Å^{-1})
Sapphire.trm	Sapphire (Al_2O_3) filter for fast neutrons ($k > 6$ Å^{-1})

Table 4.5.: Some transmission data files to be used with the Filter_gen component.

Identification

- **Author:** Thomas Kittelmann
- **Origin:** ESS
- **Date:** 2026

Description

A filter or window of an isotropic material, which attenuates the beam according to the total cross section (scattering plus absorption) of the material, as provided by the NCrystal library. Neutrons are not scattered: the weight of a neutron travelling a distance L through the material is multiplied by exp(-sigma*L), where sigma is the macroscopic cross section at its wavelength. This is suitable for e.g. Be filters or Al windows, where the scattered neutrons are not of interest. For the full NCrystal physics, use NCrystal_sample or the Union components instead.

The cross sections are tabulated during initialisation, so the component can also be used on GPUs. The table is provided by NCrystal itself, unless the C API function ncrystal_filtertable is not available, in which case the component will create the table itself as a fallback workaround.

The material is given by an NCrystal cfg-string, e.g. "stdlib::Be_sg194.ncmat;temp=80K" for a cooled Be filter, or "mycrystal.ncmat" for a local file. Oriented materials such as single crystals are not supported, but a fast sapphire filter can be emulated with an assumption that no reflections satisfy the Bragg condition: "stdlib::Al2O3_sg167_Corundum.ncmat;bragg=0;temp=200K Multiphase materials are also supported: "phases<0.98*stdlib::Al_sg225.ncmat&0.01*stdlib::Fe_sg229_Iron-alpha.ncmat&0.01*stdlib::Si_sg227.ncmat>;temp=180K".

The geometry is a box (xwidth, yheight, zdepth) or a cylinder (radius, yheight) along the y axis.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
cfg	str	NCrystal material configuration string (details on this page).	"void"
xwidth	m	x-dimension (width) of the filter, for a box	0
yheight	m	y-dimension (height) of the filter, for a box or cylinder	0
zdepth	m	z-dimension (depth) of the filter, for a box	0
radius	m	radius of the filter, for a cylinder	0

Links

- Component source code found in file `NCrystal_filter.comp`.
- The NCrystal wiki at https://github.com/mctools/ncrystal/wiki.

4.6. The `Collimator_linear` McStas Component

A simple analytical Soller collimator (with triangular transmission).

Identification

- **Author:** Kristian Nielsen
- **Origin:** Risoe
- **Date:** August 1998

Description

Soller collimator with rectangular opening and specified length. The transmission function is an average and does not utilize knowledge of the actual neutron trajectory. A zero divergence disables collimation (then the component works as a double slit).

Example: Collimator_linear(xmin=-0.1, xmax=0.1, ymin=-0.1, ymax=0.1, length=0.25, divergence=40,transmission=0.7)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xmin	m	Lower x bound on slits	-0.02
xmax	m	Upper x bound on slits	0.02
ymin	m	Lower y bound on slits	-0.05
ymax	m	Upper y bound on slits	0.05
xwidth	m	Width of slits	0
yheight	m	Height of slits	0
length	m	Distance between input and output slits	0.3
divergence	minutes of arc	Divergence horizontal angle (calculated as atan(d/length), where d is the blade spacing)	40
transmission	1	Transmission of Soller (0<=t<=1)	1
divergenceV	minutes of arc	Divergence vertical angle	0

Links

- Component source code found in file `Collimator_linear.comp`.

The simple Soller blade collimator

Collimator_linear models a standard linear Soller blade collimator. The collimator has two identical rectangular openings, defined by the x and y values. Neutrons not clearing both openings are ABSORB'ed. The length of the collimator blades is denoted L, while the distance between blades is called d.

The collimating effect is taken care of by employing an approximately triangular transmission through the collimator of width (FWHM) δ, which is given in arc minutes, *i.e.* $\delta = 60$ is one degree. If $\delta = 0$, the collimating effect is disabled, so that the component only consists of two rectangular apertures.

For a more detailed Soller collimator simulation, taking every blade into account, it is possible to use **Channeled_guide** with absorbing walls, see section 5.1.

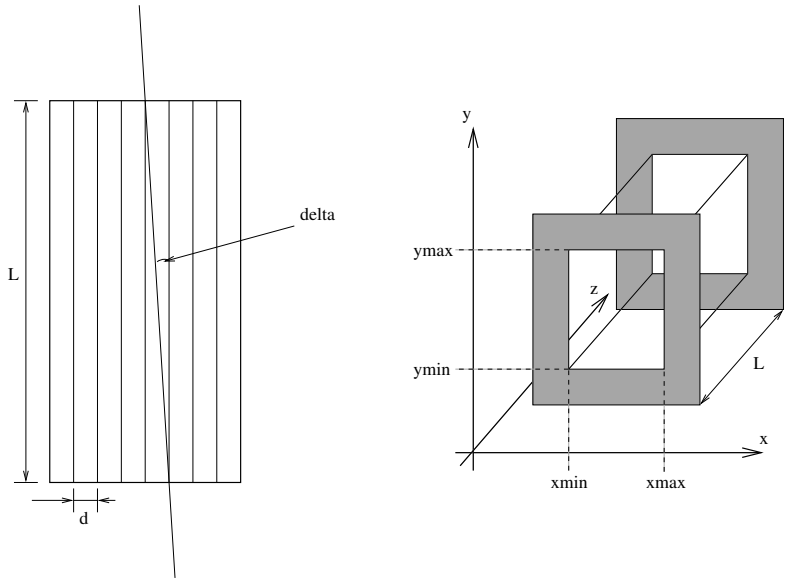


Figure 4.1.: The geometry of a simple Soller blade collimators: The real Soller collimator, seen from the top (left), and a sketch of the component **Soller** (right). The symbols are defined in the text.

Collimator transmission

The horizontal divergence, η_h, is defined as the angle between the neutron path and the vertical $y - z$ plane along the collimator axis. We then define the collimation angle as the maximal allowed horizontal divergence: $\delta = \tan^{-1}(d/L)$, see Fig. 4.1. Neutrons with a horizontal divergence angle $|\eta_h| \geq \delta$ will always hit at least one collimator blade and will thus be ABSORB'ed. For smaller divergence angles, $|\eta_h| < \delta$, the fate of the neutron depends on its exact entry point. Assuming that a typical collimator has many blades, the absolute position of each blade perpendicular to the collimator axis is thus

mostly unimportant. A simple statistical consideration now shows that the transmission probability is $T = 1 - \tan|\eta_h|/\tan\delta$. Often, the approximation $T \approx 1 - |\eta_h|/\delta$ is used, giving a triangular transmission profile.

Algorithm

The algorithm of Collimator_linear is roughly as follows:

1. Check by propagation if the neutron ray clear the entry and exit slits, otherwise ABSORB.
2. Check if $|\eta_h| < \delta$, otherwise ABSORB.
3. Simulate the collimator transmission by a weight transformation:

$$\pi_i = T = 1 - \tan|\eta_h|/\tan\delta, \tag{4.1}$$

4.7. The `Collimator_radial` McStas Component

A radial Soller collimator.

Identification

- **Author:** Emmanuel Farhi <farhi@ill.fr>
- **Origin:** ILL
- **Date:** July 2005

Description

Radial Soller collimator with rectangular opening and specified length. The collimator is made of many rectangular channels stacked radially. Each channel is a set of transmitting layers (nslit), separated by an absorbing material (infinitely thin), the whole stuff is inside an absorbing housing.

When specifying the number of channels (nchan), each channel has a total entrance width=radius*fabs(theta_max-theta_min)/nchan, but only the central portion 'xwidth' accepts neutrons. When xwidth=0, it is set to the full apperture so that all neutrons enter the channels (all walls are infinitely thin).

When using zero as the number of channels (nchan), the collimator is continuous, whithout shadowing effect.

The component should be positioned at the radius center. The component can be made oscillating (usual on diffractometers and TOF machines) with the 'roc' parameter. The neutron beam outside the collimator angular area is transmitted unaffected.

When used as a focusing collimator, the focusing parameter should be set to 1.

An example of a instrument that uses this collimator can be found in the SALSA instrument, in the example folder

Example: Channelled radial collimator with shadow parts Collimator_radial(xwidth=0.015, yheight=.3, length=.35, divergence=40,transmission=1, theta_min=5, theta_max=165, nchan=128, radius=0.9) A continuous radial collimator Collimator_radial(yheight=.3, length=.35, divergence=40,transmission=1, theta_min=5, theta_max=165, radius=0.9)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xwidth	m	Soller window width, filled with nslit slits. Use 0 value for continuous collimator.	0
yheight	m	Collimator height. If yheight_inner is specified, then this is the outer cylinders height	.3
length	m	Length/Distance between inner and outer slits.	.35
divergence	min of arc	Divergence angle. May also be specified with the nslit parameter. A zero value unactivates component.	0
transmission	1	Maximum transmission of Soller (0<=t<=1).	1
theta_min	deg	Minimum Theta angle for the radial setting.	5
theta_max	deg	Maximum Theta angle for the radial setting.	165
nchan	1	Number of Soller channels in the theta range. Use 0 value for continuous collimator.	0
radius	m	Radius of the collimator (to entry window).	1.3
nslit	1	Number of blades composing each Soller. Overrides the divergence parameter.	0
roc	deg	Amplitude of oscillation of collimator. 0=fixed.	0
verbose		Gives additional information.	0
approx		Use Soller triangular transmission approximation.	0
focusing	1	When set allows you to use the collimators for focusing, rather than dispersing.	0
yheight_inner	1	Defines the inner height of the collimator	0

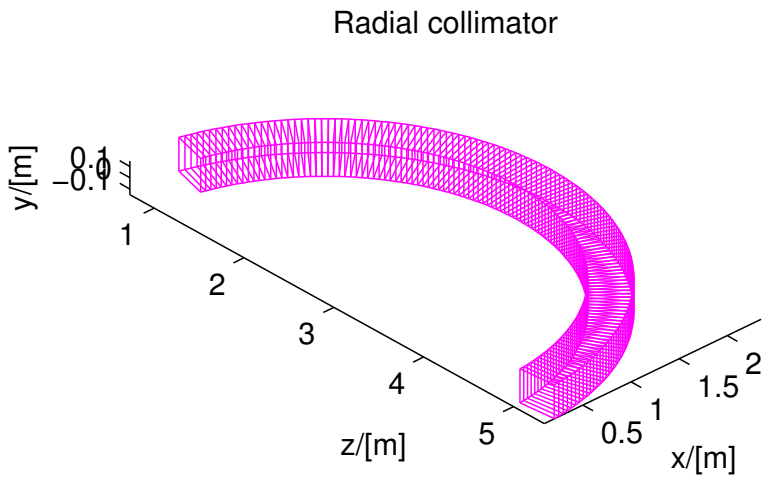


Figure 4.2.: A radial collimator

Links

- Component source code found in file `Collimator_radial.comp`.

A radial Soller blade collimator

Name:	Collimator_radial
Author:	(System) E.Farhi, ILL
Input parameters	w_1, h_1, w_2, h_2, *len*, θ_{min}, θ_{max}, *nchan*, *radius*
Optional parameters	*divergence*, *nblades*, *roc* and others
Notes	Validated

This radial collimator works either using an analytical approximation like **Collimator_linear** (see section 4.6), or with an exact model.

The input parameters are the inner radius *radius*, the radial length *len*, the input and output window dimensions w_1, h_1, w_2, h_2, the number of Soller channels *nchan* (each of them being a single linear collimator) covering the angular interval $[\theta_{min}, \theta_{max}]$ angle with respect to the z-axis.

If the *divergence* parameter is defined, the approximation level is used as in Collimator_linear (see section 4.6). On the other hand, if you perfer to describe exactly the number of blades *nblades* assembled to build a single collimator channel, then the model is exact, and traces the neutron trajectory inside each Soller. The computing efficiency is then lowered by a factor 2.

The component can be made oscillating with an amplitude of *roc* times $\pm w_1$, which supresses the channels shadow.

As an alternative, you may use the **Exact_radial_coll** contributed component.

For a rectangular shaped collimator, instead of cylindrical/radial, you may use the Guide_channeled and the Guide_gravity components.

5. Reflecting optical components: mirrors, and guides

This section describes advanced neutron optical components such as supermirrors and guides as well as various rotating choppers. A description of the reflectivity of a supermirror is found in section 5.1.

5.1. The `Guide` McStas Component

Neutron guide.

Identification

- **Author:** Kristian Nielsen
- **Origin:** Risoe
- **Date:** September 2 1998

Description

Models a rectangular guide tube centered on the Z axis. The entrance lies in the X-Y plane. For details on the geometry calculation see the description in the McStas reference manual. The reflectivity profile may either use an analytical mode (see Component Manual) or a 2-columns reflectivity free text file with format

```
[q(Angs-1) R(0-1)].

Example: Guide(w1=0.1, h1=0.1, w2=0.1, h2=0.1, l=2.0, R0=0.99, Qc=0.021, alpha=6.07, m=2, W=0.0
```

%VALIDATION May 2005: extensive internal test, no bugs found Validated by: K. Lieutenant
%BUGS This component does not work with gravitation on. Use component Guide_gravity then.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
reflect	str	Reflectivity file name. Format <q(Å-1) R(0-1)>	0
w1	m	Width at the guide entry	
h1	m	Height at the guide entry	
w2	m	Width at the guide exit	0
h2	m	Height at the guide exit	0
l	m	length of guide	
R0	1	Low-angle reflectivity	0.99
Qc	$Å^{-1}$	Critical scattering vector	0.0219
alpha	Å	Slope of reflectivity	6.07
m	1	m-value of material. Zero means completely absorbing. glass/SiO2 Si Ni Ni58 supermirror Be Diamond m= 0.65 0.47 1 1.18 2-6 1.01 1.12	2
W	$Å^{-1}$	Width of supermirror cut-off	0.003

Links

- Component source code found in file `Guide.comp`.

This section describes advanced neutron optical components such as supermirrors and guides. A description of the reflectivity of a supermirror is found in section 5.1.

Mirror: The single mirror

Name:	Mirror
Author:	System
Input parameters	l, h, m
Optional parameters	$R_0, Q_c, W, \alpha, reflect$
Notes	validated, no gravitation support

The component **Mirror** models a single rectangular neutron mirror plate. It can be used as a sample component or to *e.g.* assemble a complete neutron guide by putting multiple mirror components at appropriate locations and orientations in the instrument definition, much like a real guide is build from individual mirrors.

In the local coordinate system, the mirror lies in the first quadrant of the x-y plane, with one corner at $(0, 0, 0)$.

The input parameters of this component are the rectangular mirror dimensions (l, h) and the values of R_0, m, Q_c, W, and α for the mirror reflectivity. As a special case, if $m = 0$ then the reflectivity is zero for all Q, *i.e.* the surface is completely absorbing.

This component may produce wrong results with gravitation.

Mirror reflectivity

To compute the reflectivity of the supermirrors, we use an empirical formula derived from experimental data [Cla+98], see Fig. 5.1. The reflectivity is given by

$$R = \begin{cases} R_0 & \text{if } Q \leq Q_\mathrm{c} \\ \frac{1}{2}R_0(1-\tanh[(Q-mQ_\mathrm{c})/W])(1-\alpha(Q-Q_\mathrm{c})) & \text{if } Q > Q_\mathrm{c} \end{cases} \tag{5.1}$$

Here Q is the length of the scattering vector (in $\mathrm{\AA}^{-1}$) defined by

$$Q = |\mathbf{k_i} - \mathbf{k_f}| = \frac{m_\mathrm{n}}{\hbar}|\mathbf{v_i} - \mathbf{v_f}|, \tag{5.2}$$

m_n being the neutron mass. The number m in (5.1) is a parameter determined by the mirror materials, the bilayer sequence, and the number of bilayers. As can be seen, $R = R_0$ for $Q < Q_\mathrm{c}$, where Q_c is the critical scattering wave vector for a single layer of the mirror material. At higher values of Q, the reflectivity starts falling linearly with a slope α until a "soft cut-off" at $Q = mQ_\mathrm{c}$. The width of this cut-off is denoted W. See the example reflection curve in figure 5.1.

It is **important** to notice that when $m < 1$, the reflectivity remains constant at $R = R_0$ up to $q = Qc$, and *not* $m.Q_c$. This means that $m < 1$ parameters behave like $m = 1$ materials.

Alternatively, the Mirror, Guide and Guide_gravity components may use a reflectivity table $reflect$, which 1st column is q [$^{-1}$] and 2nd column as the reflectivity R in [0-1]. For this purpose, we provide $m = 2$ and $m = 3$ reflectivity files from SwissNeutronics (`supermirror_m2.rfl` and `supermirror_m3.rfl` in `MCSTAS/lib/data/`).

Algorithm

The function of the component can be described as

1. Propagate the neutron ray to the plane of the mirror.
2. If the neutron trajectory intersects the mirror plate, it is reflected, otherwise it is left untouched.
3. Reflection of the incident velocity $\mathbf{v}_\mathrm{i} = (v_x, v_y, v_z)$ gives the final velocity $\mathbf{v}_\mathrm{f} = (v_x, v_y, -v_z)$.
4. Calculate $Q = 2m_\mathrm{n}v_z/\hbar$.
5. The neutron weight is adjusted with the amount $\pi_i = R(Q)$.
6. To avoid spending large amounts of computation time on very low-weight neutrons, neutrons for which the reflectivity is lower than about 10^{-10} are ABSORB'ed.

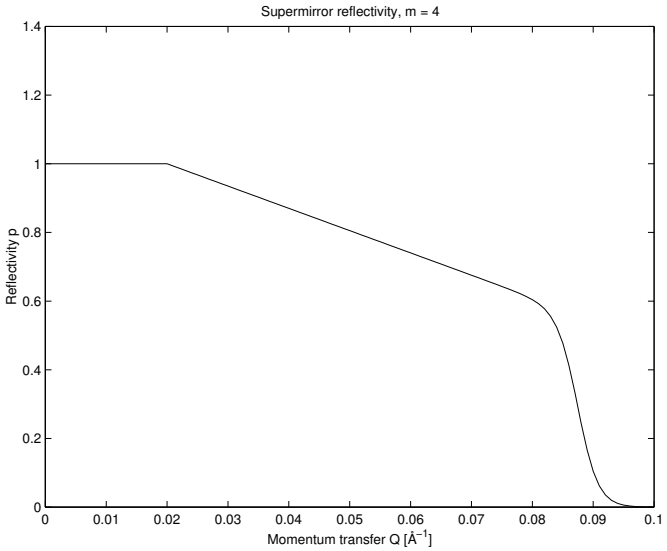


Figure 5.1.: A typical reflectivity curve for a supermirror, Eq. (5.2). The used values are $m = 4$, $R_0 = 1$, $Q_c = 0.02$ Å^{-1}, $\alpha = 6.49$ Å, $W = 1/300$ Å^{-1}.

Guide: The guide section

The component **Guide** models a guide tube consisting of four flat mirrors. The guide is centered on the z axis with rectangular entrance and exit openings parallel to the x-y plane. The entrance has the dimensions (w_1, h_1) and placed at $z = 0$. The exit is of dimensions (w_2, h_2) and is placed at $z = l$ where l is the guide length. See figure 5.2. The reflecting properties are given by the values of R_0, m, Q_c, W, and α, as for **Mirror**, or alternatively from the reflectivity file *reflect*.

Guide may produce wrong results with gravitation support. Use **Guide_gravity** (section 5.1) in this case, or the **Guide_channeled** in section 5.1.

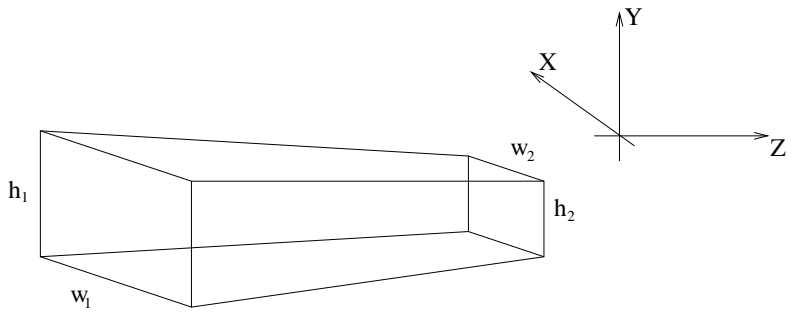


Figure 5.2.: The geometry used for the guide component.

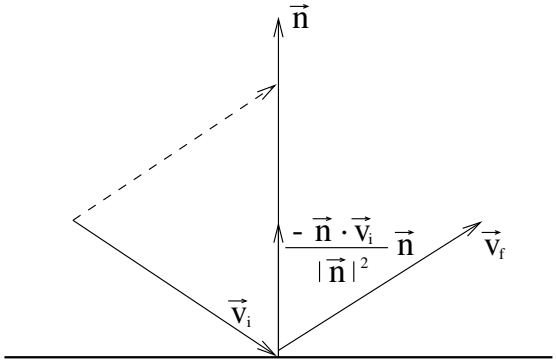


Figure 5.3.: Neutron reflecting from mirror. $\mathbf{v}_\mathrm{i}$ and $\mathbf{v}_\mathrm{f}$ are the initial and final velocities, respectively, and $\mathbf{n}$ is a vector normal to the mirror surface.

Guide geometry and reflection

For computations on the guide geometry, we define the planes of the four guide sides by giving their normal vectors (pointing into the guide) and a point lying in the plane:

$$\begin{array}{lcllcl} \mathbf{n}_1^v &=& (l, 0, (w_2 - w_1)/2) & \mathbf{O}_1^v &=& (-w_1/2, 0, 0) \\ \mathbf{n}_2^v &=& (-l, 0, (w_2 - w_1)/2) & \mathbf{O}_2^v &=& (w_1/2, 0, 0) \\ \mathbf{n}_1^h &=& (0, l, (h_2 - h_1)/2) & \mathbf{O}_1^h &=& (0, -h_1/2, 0) \\ \mathbf{n}_2^h &=& (0, -l, (h_2 - h_1)/2) & \mathbf{O}_2^h &=& (0, h_1/2, 0) \end{array}$$

In the following, we refer to an arbitrary guide side by its origin $\mathbf{O}$ and normal $\mathbf{n}$.

With these definitions, the time of intersection of the neutron with a guide side can be computed by considering the projection onto the normal:

$$t_\beta^\alpha = \frac{(\mathbf{O}_\beta^\alpha - \mathbf{r}_0) \cdot \mathbf{n}_\beta^\alpha}{\mathbf{v} \cdot \mathbf{n}_\beta^\alpha}, \tag{5.3}$$

where α and β are indices for the different guide walls, assuming the values (h,v) and (1,2), respectively. For a neutron that leaves the guide directly through the guide exit we have

$$t_\mathrm{exit} = \frac{l - z_0}{v_z} \tag{5.4}$$

The reflected velocity $\mathbf{v}_\mathrm{f}$ of the neutron with incoming velocity $\mathbf{v}_\mathrm{i}$ is computed by the formula

$$\mathbf{v}_\mathrm{f} = \mathbf{v}_\mathrm{i} - 2\mathbf{n} \cdot \frac{\mathbf{v}_\mathrm{i}}{|\mathbf{n}|^2}\mathbf{n} \tag{5.5}$$

This expression is arrived at by again considering the projection onto the mirror normal (see figure 5.3). The reflectivity of the mirror is taken into account as explained in section 5.1.

Algorithm

1. The neutron is initially propagated to the $z = 0$ plane of the guide entrance.
2. If it misses the entrance, it is ABSORB'ed.
3. Otherwise, repeatedly compute the time of intersection with the four mirror sides and the guide exit.
4. The smallest positive t thus found gives the time of the next intersection with the guide (or in the case of the guide exit, the time when the neutron leaves the guide).
5. Propagated the neutron ray to this point.
6. Compute the reflection from the side.
7. Update the neutron weight factor by the amount $\pi_i = R(Q)$.
8. Repeat this process until the neutron leaves the guide.

There are a few optimizations possible here to avoid redundant computations. Since the neutron is always inside the guide during the computations, we always have $(\mathbf{O} - \mathbf{r}_0) \cdot \mathbf{n} \leq 0$. Thus $t \leq 0$ if $\mathbf{v} \cdot \mathbf{n} \geq 0$, so in this case there is no need to actually compute t. Some redundant computations are also avoided by utilizing symmetry and the fact that many components of $\mathbf{n}$ and $\mathbf{O}$ are zero.

Guide_channeled: A guide section component with multiple channels

Name:	Guide_channeled
Author:	System
Input parameters	$w_1, h_1, w_2, h_2, l, k, m_x, m_y$
Optional parameters	$d, R_0, Q_{cx}, Q_{cy}, W, \alpha_x, \alpha_y$
Notes	validated, no gravitation support

The component **Guide_channeled** is a more complex variation of **Guide** described in the previous section. It allows the specification of different supermirror parameters for the horizontal and vertical mirrors, and also implements guides with multiple channels as used in neutron bender devices. By setting the m value of the supermirror coatings to zero, nonreflecting walls are simulated; this may be used for a very detailed simulation of a Soller collimator, see section 4.6.

The input parameters are w_1, h_1, w_2, h_2, and l to set the guide dimensions as for **Guide** (entry window, exit window, and length); k to set the number of channels; d to set the thickness of the channel walls; and R_0, W, Q_{cx}, Q_{cy}, α_x, α_y, m_x, and m_y to set the supermirror parameters as described under **Guide** (the names with x denote the vertical mirrors, and those with y denote the horizontal ones).

Algorithm

The implementation is based on that of **Guide**.

1. Calculate the channel which the neutron will enter.
2. Shift the x coordinate so that the channel can be simulated as a single instance of the **Guide** component.
3. (do the same as in **Guide**.)
4. Restore the coordinates when the neutron exits the guide or is absorbed.

Known problems

- This component may produce wrong results with gravitation support. Use Guide_gravity (section 5.1) in this case.
- The focusing channeled geometry (for $k > 1$ and different values of w_1 and w_2) is buggy (wall slopes are not computed correctly, and the component 'leaks' neutrons).

Guide_gravity: A guide with multiple channels and gravitation handling

Name:	Guide_gravity
Author:	System
Input parameters	w_1, h_1, w_2, h_2, l, k, m
Optional parameters	d, R_0, Q_c, W, α, wavy, chamfers, k_h, n, G
Notes	validated, **with** gravitation support, rotating mode

This component is a variation of **Guide_channeled** (section 5.1) with the ability to handle gravitation effects and functional channeled focusing geometry. Channels can be specified in two dimensions, producing a 2D array (k, k_h) of smaller rectangular guide channels.

The coating is specified as for the Guide and Mirror components by mean of the parameters R_0, m, Q_c, W, and α, or alternatively from the reflectivity file $reflect$.

Waviness effects, supposed to be randomly distributed (*i.e.* non-periodic waviness) can be specified globally, or for each part of the guide section. Additionally, chamfers may be defined the same way. Chamfers originate from the substrate manufacturing, so that operators do not harm themselves with cutting edges. Usual dimensions are about tens of millimeters. They are treated as absorbing edges around guide plates, both on the input and output surfaces, but also aside each mirror.

The straight section of length l may be divided into n bits of same length within which chamfers are taken into account.

The component has also the capability to rotate at a given frequenccy in order to approximate a Fermi Chopper, including phase shift. The approximation resides in the fact that the component is considered fixed during neutron propagation inside slits. Beware that this component is then located at its entry window (not centered as the other Fermi choppers).

To activate gravitation support, either select the McStas gravitation support (`mcrun --gravitation`
or from the Run dialog of `mcgui`), or set the gravitation field strength G (e.g. -9.81 on Earth).

This component is about 50 % slower than the `Guide` component, but has much more capabilities.

A contributed version Guide_honeycomb of this component exists with a honeycomb geometry.

Bender: a bender model (non polarizing)

Name:	Bender
Author:	Philipp Bernhardt
Input parameters	r, W_{in}, l, w, h
Optional parameters	$k, d, R_{0[a,i,s]}, \alpha_{[a,i,s]}, m_{[a,i,s]}, Q_{c[a,i,s]}, W_{[a,i,s]}$
Notes	partly validated, no gravitation support

The Bender component is simulating an ideal curved neutron guide (bender). It is bent to the negative X-axis and behaves like a parallel guide in the Y axis. Opposite

curvature may be achieved by a $(0, 0, 180)$ rotation (along Z-axis).

Bender radius r, entrance width w and height h are required parameters. To define the length, you may either enter the deviation angle W_{in} or the length l. Three different reflectivity profiles R_0, Q_c, W, m, α can be given (see section 5.1): for outer walls (index a), for inner walls (index i) and for the top and bottom walls (index s).

To get a better transmission coefficient, it is possible to split the bender into k channels which are separated by partitions with the thickness of d. The partitioning walls have the same coating as the exterior walls.

Because the angle of reflection doesn't change, the routine calculates the reflection coefficent for the concave and, if necessary, for the convex wall only onces, together with the number of reflections. Nevertheless the exact position, the time, and the divergence is calculated at the end of the bender, so there aren't any approximations.

The component is shown *straight* on geometrical views (mcdisplay/Trace), and the next component may be placed directly at distance $r.W_{in} = l$ *without* rotation.

Results have been compared succesfully with analytical formula in the case of an ideal reflection and cross-checked with the program `haupt`.

An other implementation of the Bender is available as the contributed component Guide_curved.

Curved guides

Real curved guides are usually made of many straight elements (about 1 m long) separated with small gaps (e.g. 1 mm). Sections of about 10 m long are separated with bigger gaps for accessibility and pumping purposes.

We give here an example description of such a section. Let us have a curved guide of total length L, made of n elements with a curvature radius R. Gaps of size d separate elements from each other. The rotation angle of individual straight guide elements is $\alpha_z = (L + d)/R * 180/\pi$ in degrees.

In order to build an independent curved guide section, we define `Arm` components at the begining and end of it.

```
COMPONENT CG_In = Arm() AT (...)

COMPONENT CG_1  = Guide_gravity(l=L/n, ...)
AT (0,0,0) RELATIVE PREVIOUS

COMPONENT CG_2  = Guide_gravity(l=L/n, ...)
AT (0,0,L/n+d) RELATIVE PREVIOUS
ROTATED (0, (L/n+d)/R*180/PI, 0) RELATIVE PREVIOUS
...
COMPONENT CG_Out = Arm() AT (0,0,L/n) RELATIVE PREVIOUS
```

The `Guide` component should be duplicated n times by copy-paste, but changing the instance name, e.g. CG_1, CG_2, ..., CG_n. This may be automated with the `COPY` or the `JUMP ITERATE` mechanisms (see User manual).

An implementation of a continuous curved guide has been contributed as component Guide_curved.

5.2. The `Guide_gravity` McStas Component

Neutron straight guide with gravity. Can be channeled and focusing. Waviness may be specified, as well as side chamfers (on substrate).

Identification

- **Author:** Emmanuel Farhi
- **Origin:** ILL (France).
- **Date:** Aug 03 2001

Description

Models a rectangular straight guide tube centered on the Z axis, with gravitation handling. The entrance lies in the X-Y plane. The guide can be channeled (nslit,d,nhslit parameters). The guide coating specifications may be entered via different ways (global, or for each wall m-value).

Waviness (random) may be specified either globally or for each mirror type. Side chamfers (due to substrate processing) may be specified the same way. In order to model a realistic straight guide assembly, a long guide of length 'l' may be splitted into 'nelements' between which chamfers/gaps are positioned.

The reflectivity may be specified either using an analytical mode (see Component manual) or as a text file in free format, with 1st column as q[Å-1] and 2nd column as the reflectivity R in [0-1]. For details on the geometry calculation see the description in the McStas component manual.

There is a special rotating mode in order to approximate a Fermi Chopper behaviour, in the case the neutron trajectory is nearly linear inside the chopper slits, i.e. the neutrons are fast w/r/ to the chopper speed. Slits are straight, but may be super-mirror coated. In this case, the component is NOT centered, but located at its entry window. It should then be shifted by -l/2.

Example: Guide_gravity(w1=0.1, h1=0.1, w2=0.1, h2=0.1, l=12,

```
R0=0.99, Qc=0.0219, alpha=6.07, m=1.0, W=0.003)
```

%VALIDATION May 2005: extensive internal test, all problems solved Validated by: nslit. Lieutenant

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
w1	m	Width at the guide entry	

Name	Unit	Description	Default
h1	m	Height at the guide entry	
w2	m	Width at the guide exit. If 0, use w1.	0
h2	m	Height at the guide exit. If 0, use h1.	0
l	m	length of guide	
R0	1	Low-angle reflectivity	0.995
Qc	$Å^{-1}$	Critical scattering vector	0.0218
alpha	Å	Slope of reflectivity	4.38
m	1	m-value of material. Zero means completely absorbing. m=0.65 glass/SiO2 Si Ni Ni58 supermirror Be Diamond m= 0.65 0.47 1 1.18 2-6 1.01 1.12 for glass/SiO2, m=1 for Ni, 1.2 for Ni58, m=2-6 for supermirror. m=0.47 for Si	1.0
W	$Å^{-1}$	Width of supermirror cut-off	0.003
nslit	1	Number of vertical channels in the guide (>= 1) (nslit-1 vertical dividing walls).	1
d	m	Thickness of subdividing walls	0.0005
mleft	1	m-value of material for left. vert. mirror	-1
mright	1	m-value of material for right. vert. mirror	-1
mtop	1	m-value of material for top. horz. mirror	-1
mbottom	1	m-value of material for bottom. horz. mirror	-1
nhslit	1	Number of horizontal channels in the guide (>= 1). (nhslit-1 horizontal dividing walls). this enables to have nslit*nhslit rectangular channels	1
G	m/s^2	Gravitation norm. 0 value disables G effects.	0
aleft	1	alpha-value of left vert. mirror	-1
aright	1	alpha-value of right vert. mirror	-1
atop	1	alpha-value of top horz. mirror	-1
abottom	1	alpha-value of left horz. mirror	-1
wavy	deg	Global guide waviness	0
wavy_z	deg	Partial waviness along propagation axis	0
wavy_tb	deg	Partial waviness in transverse direction for top/bottom mirrors	0
wavy_lr	deg	Partial waviness in transverse direction for left/right mirrors	0
chamfers	m	Global chamfers specifications (in/out/mirror sides).	0
chamfers_z	m	Input and output chamfers	0

Name	Unit	Description	Default
chamfers_lr	m	Chamfers on left/right mirror sides	0
chamfers_tb	m	Chamfers on top/bottom mirror sides	0
nelements	1	Number of sections in the guide (length l/nelements).	1
nu	Hz	Rotation frequency (round/s) for Fermi Chopper approximation	0
phase	deg	Phase shift for the Fermi Chopper approximation	0
reflect	str	Reflectivity file name. Format <q(Å-1) R(0-1)>	"NULL"

Links

- Component source code found in file `Guide_gravity.comp`.

6. Moving optical components: Choppers and velocity selectors

We list in this chapter some moving optical components, like choppers, that may be used for TOF class instrument simulations, and velocity selector used for partially monochromatize continuous beams.

6.1. The `DiskChopper` McStas Component

Based on Chopper (Philipp Bernhardt), Jitter and beamstop from work by Kaspar Hewitt Klenoe (jan 2006), adjustments by Rob Bewey (march 2006)

Identification

- **Author:** Peter Willendrup
- **Origin:** Risoe
- **Date:** March 9 2006

Description

Models a disc chopper with nslit identical slits, which are symmetrically distributed on the disc. At time t=0, the centre of the first slit opening will be situated at the vertical axis when phase=0, assuming the chopper centre of rotation is placed **BELOW** the beam axis. If you want to place the chopper **ABOVE** the beam axis, please use a 180 degree rotation around Z (otherwise unexpected beam splitting can occur in combination with the isfirst=1 setting, see related bug on GitHub)

For more complicated gemometries, see component manual example of DiskChopper GROUPing.

If the chopper is the 1st chopper of a continuous source instrument, you should use the "isfirst" parameter. This parameter SETS the neutron time to match the passage of the chooper slit(s), taking into account the chopper timing and phasing (thus conserving your simulated statistics).

The isfirst parameter is ONLY relevant for use in continuous source settings.

Example: DiskChopper(radius=0.2, theta_0=10, nu=41.7, nslit=3, delay=0, isfirst=1) First chopper DiskChopper(radius=0.2, theta_0=10, nu=41.7, nslit=3, delay=0, isfirst=0)

NOTA BENE wrt. GROUPing and isfirst: When setting up a GROUP of DiskChoppers for a steady-state / reactor source, you will need to set up 1) An initial chopper with isfirst=1, NOT part of the GROUP - and using a "big" chopper opening that spans the full angular extent of the openings of the subsequent GROUP 2) Add your DiskChopper GROUP setting isfirst=0

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
theta_0	deg	Angular width of the slits.	0
radius	m	Radius of the disc	0.5
yheight	m	Slit height (if = 0, equal to radius). Auto centering of beam at half height.	
nu	Hz	Frequency of the Chopper, omega=2*PI*nu (algebraic sign defines the direction of rotation)	
nslit	1	Number of slits, regularly arranged around the disk	3
jitter	s	Jitter in the time phase	0
delay	s	Time 'delay'	0
isfirst	0/1	Set it to 1 for the first chopper position in a cw source (it then spreads the neutron time distribution)	0
n_pulse	1	Number of pulses (Only if isfirst)	1
abs_out	0/1	Absorb neutrons hitting outside of chopper radius?	1
phase	deg	Angular 'delay' (overrides delay)	0
xwidth	m	Horizontal slit width opening at beam center	0
verbose	1	Set to 1 to display Disk chopper configuration	0

Links

- Component source code found in file `DiskChopper.comp`.

The disc chopper

To cut a continuous neutron beam into short pulses, or to control the pulse shape (in time) from a pulsed source, one can use a disc chopper (see figure 6.1). This is a fast rotating disc with the rotating axis parallel to the neutron beam. The disk consists of

neutron absorbing materials. To form the pulses the disk has openings through which the neutrons can pass.

Component **DiskChopper** is an infinately thin, absorbing disc of radius R with n slit openings of height h and angular width θ_0. The slits are symmetrically disposed on the disc. If unset, the slit height h will extend to the centre of the disc ($h = R$).

The **DiskChopper** is self-centering, meaning that the centre of the slit openings will automatically be positioned at the centre of the beam axis (see figure 6.1). To override this behaviour, set the paramter $compat = 1$, positioning the chopper centre at height $-R$ - as implemented in the original **Chopper** component.

Optionally, each slit can have a central, absorbing insert - a *beamstop* of angular width θ_1. For more exotic chopper definitions, use the `GROUP` keyword, see below for an example.

The direction of rotation can be controlled, which allows to simulate e.g. counter-rotating choppers. The phase or time-delay t_0 (in seconds) is defined by the time where the first of the n slits is positioned at the top. As an alternative, an angular phase can be given using the ϕ_0 parameter.

By default, neutrons hitting outside the physical extent of the disc are absorbed. This behaviour can be overruled by setting parameter $abs_out = 0$.

When simulating the chopping of a continuous beam, most of the neutrons could easily be lost. To improve efficiency, one can set the flag `IsFirst`, which will allow every neutron ray to pass the **DiskChopper**, but modify the time, t, to a (random) time at which it is possible to pass. Of course, there should be only one "first chopper" in any simulation. To simulate frame overlap from a "first chopper", one can specify the number of frames to study by the parameter n_{pulse}.

For more advanced chopper geometries than those mentioned above, it is possible to set up a `GROUP` of choppers:

```
1 COMPONENT Chop1 = DiskChopper(omega=2500, R=0.3, h=0.2, theta_0=20, n=1)
2 AT (0, 0, 1.1) RELATIVE Source
3 GROUP Choppers
4
5 COMPONENT Chop2 = DiskChopper(omega=2500, R=0.3, h=0.2, theta_0=20, n=1,
6                               phi_0=40)
7 AT (0, 0, 1.1) RELATIVE Source
8 GROUP Choppers
```

The result of such a **DiskChopper** `GROUP`ing can be seen in figure 6.2

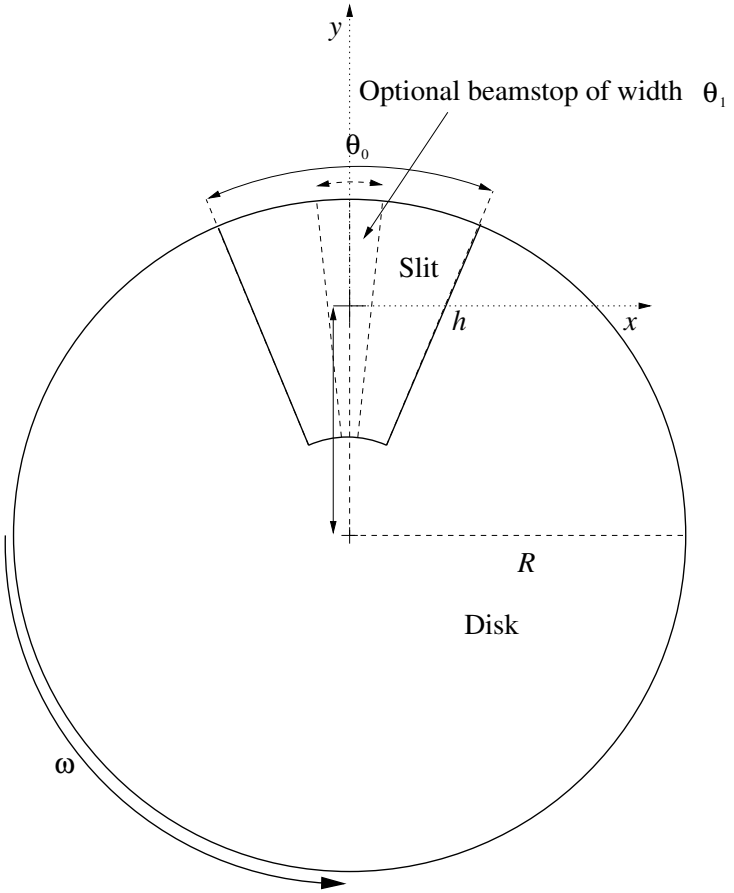


Figure 6.1.: Sketch of a disc chopper with geometry parameters

6.2. The `FermiChopper` McStas Component

Fermi Chopper with rotating frame.

Identification

- **Author:** M. Poehlmann, C. Carbogno, H. Schober, E. Farhi
- **Origin:** ILL Grenoble / TU Muenchen
- **Date:** May 2002

Description

Models a fermi chopper with optional supermirror coated blades supermirror facilities may be disabled by setting m = 0, R0=0 Slit packages are straight. Chopper slits are separated by an infinitely thin absorbing material. The effective transmission (resulting

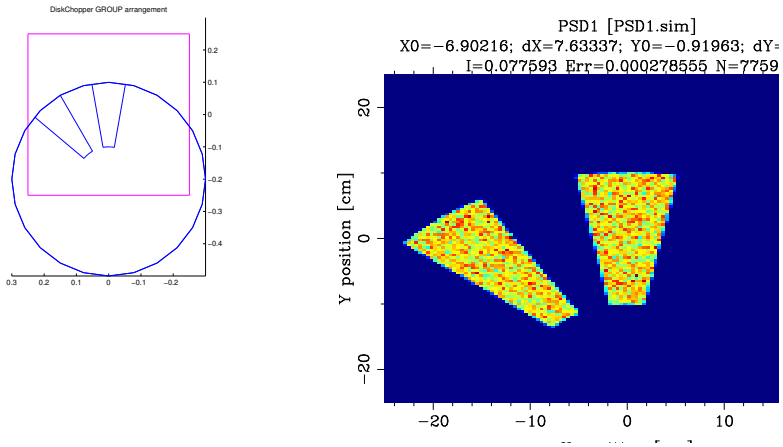


Figure 6.2.: `mcdisplay` rendering and monitor output from a DiskChopper `GROUP`

from fraction of the transparent material and its transmission) may be specified. The chopper slit package width may be specified through the total width 'xwidth' of the full package or the width 'w' of each single slit. The other parameter is calculated by: xwidth = nslit*w. The slit package may be made curved and use super-mirror coating. Neutrons that miss the chopper cylinder are transmitted unchanged, so a realistic set-up needs shielding or a beam-defining slit around it; a monitor downstream that is wider than the chopper also counts the unchopped beam.

Example:

```
FermiChopper(phase=-50.0, radius=0.04, nu=100, yheight=0.08, w=0.00022475, nslit=200.0, R0=0.0,
```

%VALIDATION Apr 2005: extensive external test, most problems solved (cf. 'Bugs')
Validated by: K. Lieutenant, E. Farhi

limitations: no absorbing blade width used

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
phase	deg	chopper phase at t=0	0
radius	m	chopper cylinder radius	0.04

Name	Unit	Description	Default
nu	Hz	chopper frequency. Omega=2*PI*nu in rad/s, nu*60 in rpm. Positive value corresponds to a counter-clockwise rotation around y.	100
w	m	width of one chopper slit	0.00022475
nslit	1	number of chopper slits	200
R0	1	low-angle reflectivity	0.0
Qc	$Å^{-1}$	critical scattering vector	0.02176
alpha	Å	slope of reflectivity	2.33
m	1	m-value of material. Zero means completely absorbing.	0.0
W	$Å^{-1}$	width of supermirror cut-off	2e-3
length	m	channel length of the Fermi chopper. The default suits the fine 200-slit package of the Example; set it for any other package (the curved packages in ISIS_MERLIN use about 0.1 m).	0.012
eff	1	efficiency = transmission x fraction of transparent material	0.95
zero_time	1	set time to zero: 0=no, 1=once per half cycle, 2=auto adjust phase	0
xwidth	m	optional total width of slit package	0
verbose	1	set to 1,2 or 3 gives debugging information	0
yheight	m	height of slit package	0.08
curvature	m^{-1}	Curvature of slits (1/radius of curvature).	0
delay	s	sets phase so that transmision is centered on 'delay'	0

Links

- Component source code found in file `FermiChopper.comp`.
- Vitess_ChopperFermi component by
- G. Zsigmond, imported from Vitess by K. Lieutenant.

The Fermi-chopper

The chopper geometry and parameters

The Fermi chopper is a rotating vertical cylinder containing a set of collimating slits (*slit package*). Main geometry parameters are the radius R, minimum and maximum height

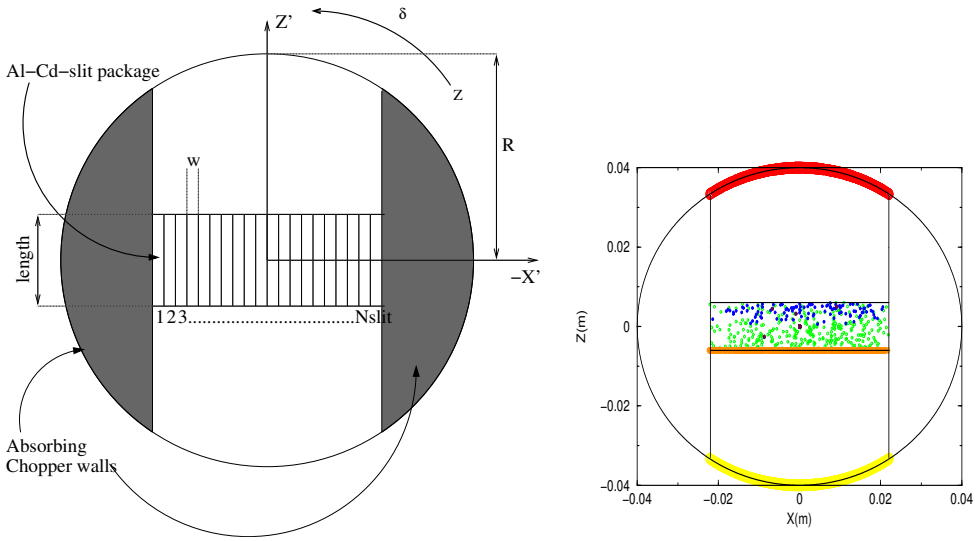


Figure 6.3.: Geometry of the Fermi-chopper (left) and Neutrons in the chopper (right).

y_{min} and y_{max} (see Fig. 6.3). In this implementation, the slits are by default straight, but may be coated with super-mirror, and curved. Main parameters for the slits are the number of slits $Nslit$, the length *length* and width w of each slit, the width of the separating Cd-blades is neglected. The slit walls reflectivity is modelled just like in guide components by the m-value ($m > 1$ for super mirrors), the critical scattering vector Q_c, the slope of reflectivity α, the low-angle reflectivity R_0 and the width of supermirror cut-off W. For $m = 0$ the blades are completly absorbing. The AT position of the component is its center.

The angular speed of the chopper is $\omega = 2\pi\nu$, where ν is the rotation frequency. The angle *phase* for which the chopper is in the 'open' state for most of the neutrons coming in (z' axis of the rotating frame parallel to the z axis of the static frame) is also an input parameter. The time window may optionally be shifted to zero when setting the `zero_time=1` option. A phase guess value may be set automatically using the `zero_time=2` option.

The curvature of the slit channels is specified with the *curvature* parameter. Positive sign indicates that the deviation 'bump' due to curvature is in the x' positive side, and the center of curvature is in the x' negative side. The optimal radius of curvature R is related to frequency ν and neutron velocity v with: $v = 4\pi R\nu$.

The component was validated extensively by K. Lieutenant. As an alternative, one may use the **Vitess_ChopperFermi** component (eventhough slower and without super-mirror support) or the **FermiChopper_ILL** contributed component. The Guide_gravity component has also a rotating mode, using an approximation of a Fermi Chopper.

Parameter	unit	meaning
radius	[m]	chopper cylinder radius
ymin	[m]	lower y bound of cylinder
ymax	[m]	upper y bound of cylinder
Nslit	[1]	number of chopper slits
length	[m]	channel length of the Fermi chopper
w	[m]	width of one chopper slit. May also be specified as *width*=w*Nslit for total width of slit package.
nu	[Hz]	chopper frequency
phase	[deg]	chopper phase at t=0
zero_time	[1]	shit time window around 0 if true
curvature	[m^{-1}]	Curvature of slits (1/radius of curvature)
m	[1]	slit coating parameters. See section 5.1
alpha	[Å]	
Qc	[$Å^{-1}$]	
W	[$Å^{-1}$]	
R0	[1]	

Table 6.3.: FermiChopper component parameters

Propagation in the Fermi-chopper

As can be seen in figure 6.3, neutrons first propagate onto the cylinder surface of the chopper (yellow curve). Then the program checks the interaction with the entrance of the slit package (orange line) and calculates which slit is hit. If the slit coating is reflecting ($m > 0$), multiple reflections are calculated (green, blue and maroon circles), otherwise the neutrons are absorbed as soon as they interact with the blades. Finally the remaining neutrons propagate to the exit of the chopper (red curve).

The rotation of the chopper is characterized by the angle δ between the rotating z' and the static z-axis. $\delta(t)$ is defined by:

$$\delta(t) = \widehat{z, z'} = \omega.(t - t_0) = \omega.t + \phi_0$$

where t is the absolute time, t_0 is the chopper delay, and ϕ_0 is the chopper phase. The chopper should better be *time focussing*: slow neutrons should pass before the fast ones, so that they finally hit the detectors at the same time. Therefore the signs of ω and δ are very important: For $t > t_0$, δ is positive and points anti-clockwise.

Since the rotation is applied along the y - axis, we can simplify the problem to two dimensions. The orthogonal transformation matrix T from the static (zx) to the rotating frame $(z'x')$ is:

$$T_{zx \to z'x'} = \begin{pmatrix} \cos(\delta) & \sin(\delta) \\ -\sin(\delta) & \cos(\delta) \end{pmatrix} \tag{6.1}$$

Together with the equation for a non-accelerated, linear propagation $\vec{r} = \vec{r_0} + \vec{v}t$

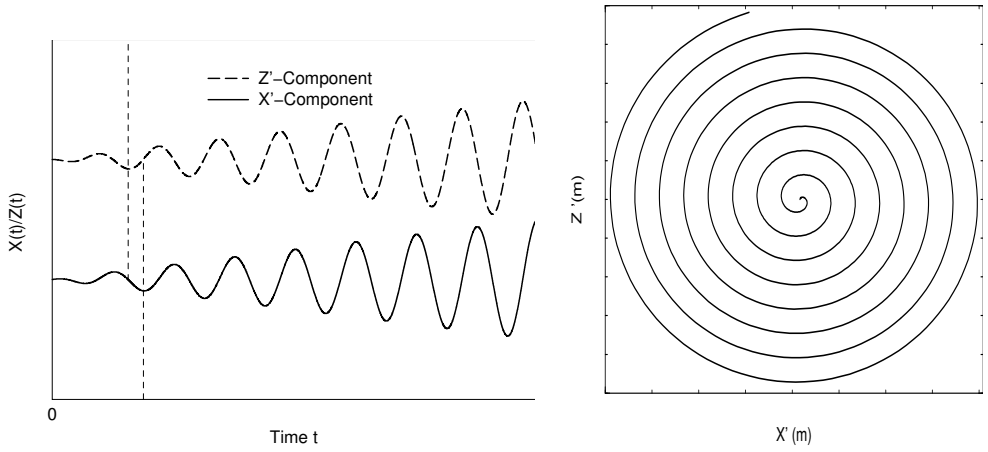


Figure 6.4.: The x' and z' component as a function of time in the rotating frame (left). A typical neutron trajectory in the rotating frame (right).

the orthogonal transformation produces a curve in the Z'-X'-plane known as *archidemic spiral*, as can be seen in figure 6.4. The two vector components $s(t) = (z', x')$ follow the equation:

$$s(t) = \begin{pmatrix} z' \\ x' \end{pmatrix} = T. \begin{pmatrix} z(t) \\ x(t) \end{pmatrix} = \begin{pmatrix} (z_0 + v_z.t)cos(\delta(t)) + (x_0 + v_x.t)sin(\delta(t)) \\ -(z_0 + v_z.t)sin(\delta(t)) + (x_0 + v_x.t)cos(\delta(t)) \end{pmatrix}. \tag{6.2}$$

For a fixed chopper rotation speed, the neutron trajectory tends to strech from a spiral curve for slow neutrons to a straight line for fast neutrons. For real Fermi chopper settings ν (about 100 Hz on IN6 at the ILL), neutron trajectories are found to be nearly straight for 1000 m/s neutron velocities [Bla83].

The basis of the algorithm is to find the intersections of these spiral trajectories with the chopper outer cylinder and then the slit package, in the rotating frame.

For this purpose, the *Ridders's* root finding method was implemented [Pre+02] in order to solve

$$x'(t) = d \text{ or } z'(t) = d \tag{6.3}$$

This method provides faster and more accurate intersection determination than other common algorithms. E.g. the secant method fails more often and may give wrong results (outside chopper) whereas the bisection method (a.k.a Picard dichotomy) is slightly slower.

Standard slit packages (non super-mirror)

The neutrons are first propagated to the outer chopper cylinder and their coordinates are transformed into the rotating frame using T. Neutrons outside the slit channel (chopper

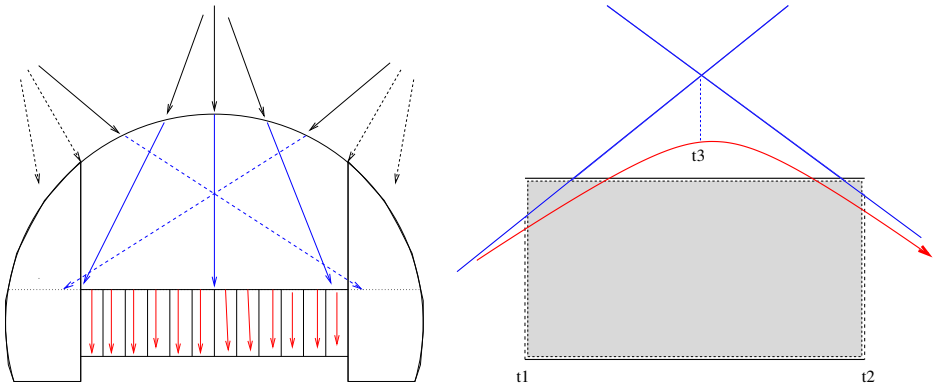


Figure 6.5.: The different steps in the algorithm (left). A neutron trajectory in a slit (right)

opening), or hitting the top and bottom caps are absorbed (yellow dots in Fig. 6.3). The side from which the neutron approaches the chopper is known (positive or negative z'-axis of the rotating frame) so that the calculation of the time of interaction with the slit package entrance t_1 is performed solving $z' = \pm\frac{\text{length}}{2}$ in Eq. (6.2). Using the result of the numerical algorithms the neutron propagates to the entrance of the slit package (orange circles in Fig. 6.3). Neutrons getting aside the slit package entrance are absorbed. Additionally, the slit package exit time t_2 is estimated the same way with $z' = \mp\frac{\text{length}}{2}$, in order to evaluate the whole time-of-flight in the chopper. The index of the slit which was hit is also computed, as we know the x' coordinate in the rotating frame at the slit entrance.

Differentiating Eq. (6.2) for x coordinate

$$\dot{x'}(t) = v'_x(t) = [v_x - \omega.(z + v_z.t)]\cos(\omega(t - t_0)) - [v_z + \omega.(x + v_x.t)]\sin(\omega(t - t_0)) \quad (6.4)$$

we may estimate the tangents to the spiral neutron trajectory in the rotating frame at times t_1 and t_2. The intersection of these two lines gives an intermediate time t_3.

If the neutron remains in the same slit at this point, then there is no intersection with the slit walls (direct flight), and the neutron may be propagated to the slit output, and then to the cylinder output. A last check is made for the neutron to pass the chopper aperture in the cylinder.

If the neutron changes of slit channel at this point, we may determine the intersection time of the neutron trajectory within $[t_1, t_3]$ or $[t_3, t_2]$, as seen in Fig. 6.5. If walls are not reflecting, we just absorb neutrons here.

The reflections (super-mirror slits)

If slit walls are reflecting, neutron is first propagated to the slit separating surface. Then the velocity in the rotating frame is computed using Eq. (6.2). Perpendicular velocity v'_x is reverted for reflection, and inverse T transformation is performed. Reflected intensity

is computed the same way as for the guide component (see section 5.1). The remaining time t_2 to the slit output is estimated and the tangent intersection process is iterated, until neutron exits. Remember that super mirror $m < 1$ parameters behave like $m = 1$ materials (see section 5.1). Selecting $m = 0$ sets the blabes absorbing.

The propagation is finalized when determining the intersection of the neutron trajectory with the outer surface of the chopper cylinder. The neutron must then pass its aperture, else it is absorbed.

WARNING: Issues have been reported for the supermirror slit option in this component. The component works correctly when using the standard, absorbing slits. We will be back with more information during the course of 2017. Meanwhile we suggest to instead use Guide_channeled with the rotate/derotate option shown in the test instrument Test_Fermi.instr.

Curved slit packages

The effect of curvature can significantly improve the flux and energy resolution shape.

As all (zx) cordinates are transformed into $(z'x')$, the most efficient way to take into account the curvature is to include it in the transformation Eq. (6.2) by 'morphing' the curved rotating real space to a straight still frame. We use parabolic curvature for slits. Then instead of solving

$$x'(t) = d - \Delta_{x'}(z') \text{ where } \Delta_{x'}(z') = R_{slit}.(1 - \sqrt{1 - (z'/R_{slit})^2}) \tag{6.5}$$

with Δ being the gap between the straight tangent line at the slit center and the real slit shape, we perform the additional transformation

$$x' \rightarrow x' + \Delta_{x'}(z') \tag{6.6}$$

The additional transformation counter-balances the real curvature so that the rest of the algorithm is written as if slits were straight. This applies to all computations in the rotating frame, and thus as well to reflections on super mirror coatings.

6.3. The `Vitess_ChopperFermi` McStas Component

Fermi chopper with absorbing walls using the VITESS module 'chopper_fermi'

Identification

- **Author:** Geza Zsigmond
- **Origin:** VITESS module 'chopper_fermi'
- **Date:** Sep 2004

Description

This component simulates a Fermi chopper with absorbing walls. The rotation axis is vertical (y-axis), i.e. the path length through the channels is given by the length 'depth' along the z-axis. The shape of the channels can be straight, curved with circular, or curved with ideal (i.e. close to a parabolic) shape. This is determined by the parameter 'GeomOption'.

Geometry for straight and circular channels: The geometry of the chopper consists of a rectangular shaped object with a channel system. In transmission position, there are 'Nchannels' slits along the x-axis, separated by absorbing walls of thickness 'wallwidth' giving a total width 'width'. The rectangular channel system is surrounded by a so-called shadowing cylinder (cf component manual).

Geometry for parabolic channels: In this case, the Fermi chopper is supposed to be a full cylinder, i.e. the central channels are longer than those on the edges (cf. figure in the component manual). The other features are the same as for the other options.

Apart from the frequency of rotation, the phase of the chopper at t=0 has to be given; phase = 0 means transmission orientation.

The option 'zerotime' may be used to reset the time at the chopper position. The consequence is that only 1 pulse is generated instead of several.

NOTE: This component must NOT be located at the same position as the previous one. This also stands for monitors and Arms. A non zero distance must be defined.

Examples: straight Fermi chopper, 18000 rpm, 20 channels a 0.9 mm separated by 0.1 mm walls, 16 mm channel length, minimal shadowing cylinder, phased to be open at 1 ms, generation of only 1 pulse, normal precision (for short wavelength neutrons) Vitess_ChopperFermi(GeomOption=0, zerotime=1, Nchannels=20, Ngates=4,

```
freq=300.0,   height=0.06, width=0.0201,
depth=0.016,  r_curv=0.0,  diameter=0.025691, Phase=-108.0,
```

wallwidth=0.0001, sGeomFileName="FC_geom_str.dat")

Fermi chopper with circular channels, 12000 rpm, optimized for 6 Å, several pulses, highest accuracy (because of long wavelength neutrons used), rest as above

```
Vitess_ChopperFermi(GeomOption=2, zerotime=0,    Nchannels=20,  Ngates=8,
freq=200.0,   height=0.06,   width=0.0201,
depth=0.016,  r_curv=0.2623, diameter=0.025691, Phase=-72.0,
```

wallwidth=0.0001, sGeomFileName="FC_geom_circ.dat")

%VALIDATION Apr 2005: extensive external test, most problems solved (cf. 'Bugs' and source header) Validated by: K. Lieutenant

limitations: slow (10 times slower than FermiChopper), especially for a high number of channels

%BUGS reduction of transmission by a large shadowing cylinder underestimated

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sGeomFileName	str	name of output file for geometry information	0
GeomOption	1	option: 0:straight 1:parabolic 2:circular	0
zerotime	1	option: 1:'set time to zero' 0: 'do not'	0
Nchannels	1	number of channels of the Fermi chopper	20
Ngates	1	number of gates defining the channel: 4=default, 6 or 8 for long wavelengths	4
freq	Hz	number of rotations per second	300.0
height	m	height of the Fermi chopper	0.05
width	m	total width of the Fermi chopper	0.04
depth	m	channel length of the Fermi chopper	0.03
r_curv	m	radius of curvature of the curved Fermi chopper	0.5
diameter	m	diameter of the shadowing cylinder	0.071
Phase	deg	dephasing angle at zero time	0.0
wallwidth	m	thickness of walls separating the channels	0.0002

Links

- Component source code found in file `Vitess_ChopperFermi.comp`.
- straight VITESS Fermi chopper
- curved VITESS Fermi chopper

The Fermi Chopper from Vitess

The component **Vitess_ChopperFermi** simulates a Fermi chopper with absorbing walls. The shape of the channels can be straight, curved with circular, or curved with ideal (i.e. close to a parabolic) shape. This is determined by the parameter 'GeomOption'. In the option 'straight Fermi chopper', the very fast neutrons are transmitted with only a time modulation and lower speed neutrons are modulated both in time of flight and wavelength. If the channels are curved, the highest transmission occurs for a wavelength

$$\lambda_{\text{opt}} = \frac{3956[\text{mÅ/s}]}{2\omega r_{\text{curv}}} \tag{6.7}$$

with

$$\omega = 2\pi f \tag{6.8}$$

The optimal shape is calculated in an exact way and is close to parabolic; in this case, transmission is as high for the optimal wavelength as in the case of a straight Fermi chopper for the limit $\lambda \to 0$. In the more realistic case of circular shapes channels, the transmission is slightly lower. In general, neutrons are transmitted through a curved Fermi chopper with a time AND wavelength modulation .

The rotation axis is vertical (y-axis), i.e. the path length through the channels is given by the length l along the z-axis. The inital orientation is given by the phase ϕ of the chopper - $\phi = 0$ means transmission orientation.

Geometry for **straight** and **circular** channels: The geometry of the chopper consists of a rectangular shaped object with a channel system. In transmission position, there are N_{gates} slits of width w_{slit} each along the x-axis, separated by absorbing walls of thickness w_{wall} (see figure 6.6). The total width w_{tot} is given by

$$w_{\text{tot}} = N_{\text{gates}} w_{\text{slit}} + (N_{\text{gates}} + 1) w_{\text{wall}} \tag{6.9}$$

The rectangular channel system is surrounded by a so-called shadowing cylinder; it is a part of a cylinder with vertical symmetry axis and diameter

$$d \geq \sqrt{l^2 + w_{\text{tot}}^2} \tag{6.10}$$

It serves to prevent transmission of neutrons which do not fly through the channels; but it also reduces the transmission, because the cylinder removes neutrons in front of the channel entrance or behind the channel exit (see figure 6.6).

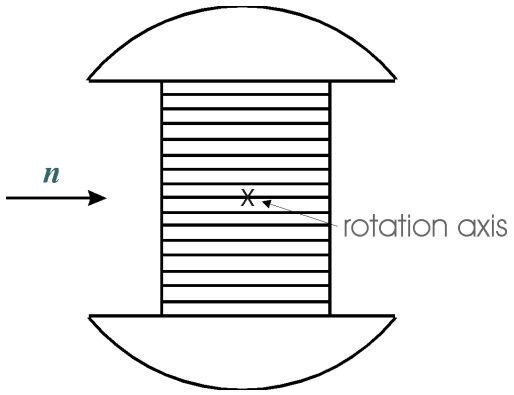


Figure 6.6.: geometry of a staight Fermi chopper

Geometry for **parabolic** channels: In this case, the Fermi chopper is supposed to be a full cylinder, i.e. the central channels are longer than those on the edges. The other features are the same as for the other options. (see figure 6.7).

The algorithm works with a rotating chopper framework. Neutrons hitting the channel walls are absorbed. The channels are approximated by N_{gates} gates. If the trajectory takes a course through all the gates, the neutron passes the Fermi chopper. There are

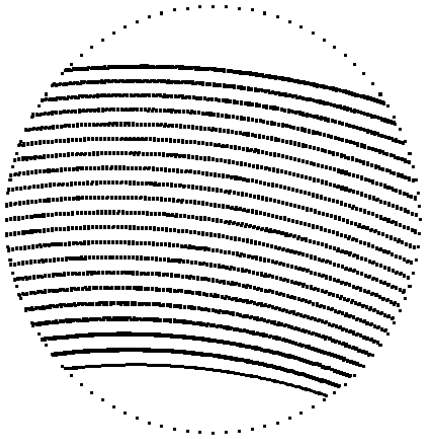


Figure 6.7.: geometry of a curved Fermi chopper

gates at the entrance and the exit of the channel. The other gates are situated close to the centre of the Fermic chopper. Precision of the simulation increases with the number of gates, but also the computing time needed. The use of four channels already gives exact transmission shapes for lower wavelengths ($\lambda < 6$ Å) and good approximation for higher ones. It is recommended to use larger number of channels only for a check.

The option 'zerotime' may be used to reset the time at the chopper position. The time is set to a value between -$T_\mathrm{p}/2$ and +$T_\mathrm{p}/2$ (with T_p being the maximal pulse length), depending on the phase of the chopper at the moment of passing the chopper centre. The result is the generation of only 1 pulse instead of several; this is useful for TOF instruments on continuous sources.

This component is about twice slower than the `FermiChopper` component.

The component must be placed after a component which sets a non zero flight path to the Fermi Chopper (e.g. not an Arm).

6.4. The `V_selector` McStas Component

Velocity selector.

Identification

- **Author:** Kim Lefmann
- **Origin:** Risoe
- **Date:** Nov 25, 1998

Description

Velocity selector consisting of rotating Soller-like blades defining a helically twisted passage. Geometry defined by two identical, centered apertures at 12 o'clock position, Origo is at the centre of the selector (input is at -zdepth/2). Transmission is analytical assuming a continuous source.

Example: V_selector(xwidth=0.03, yheight=0.05, zdepth=0.30, radius=0.12, alpha=48.298, length=0.25, d=0.0004, nu=20000, nslit=72) These are values for the D11@ILL Dornier 'Dolores' Velocity Selector (NVS 023)

%VALIDATION Jun 2005: extensive external test, no problems found Validated by: K. Lieutenant

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xwidth	m	Width of entry aperture	0.03
yheight	m	Height of entry aperture	0.05
zdepth	m	Distance between apertures, for housing containing the rotor	0.30
radius	m	Height from aperture centre to rotation axis	0.12
alpha	deg	Twist angle along the cylinder	48.298
length	m	Length of cylinder/rotor (less than zdepth)	0.25
d	m	Thickness of blades	0.0004
nu	Hz	Cylinder rotation speed, counter-clockwise, which is ideally 3956*alpha*DEG2RAD/2/PI/lambda/length	300
nslit	1	Number of Soller blades	72

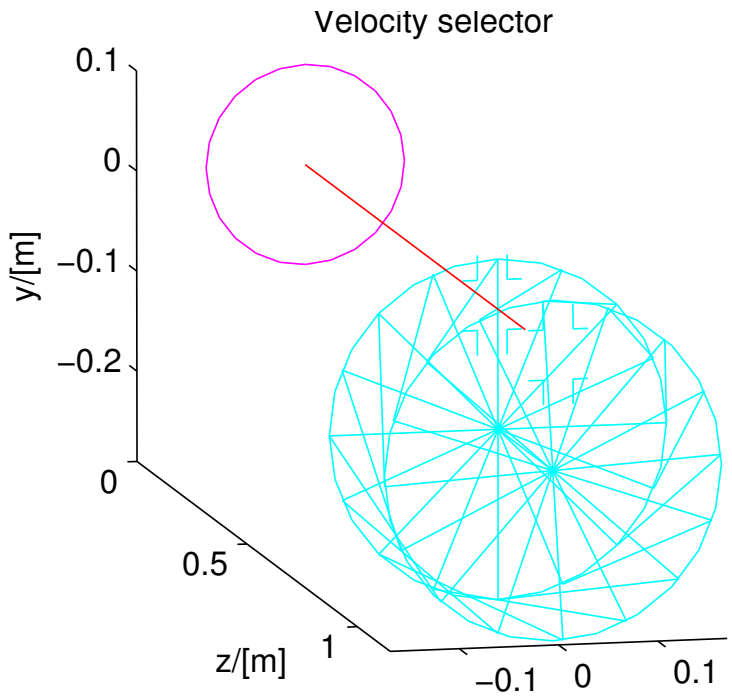


Figure 6.8.: A velocity selector

Links

- Component source code found in file `V_selector.comp`.

A rotating velocity selector

The component **V_selector** models a rotating velocity selector constructed from N collimator blades arranged radially on an axis. Two identical slits ($height \times width$) at a 12 o'clock position allow neutron passage at the position of the blades. The blades are "twisted" on the axis so that a stationary velocity selector does not transmit neutrons; the total twist angle is denoted ϕ (in degrees).

Further input parameters for **V_selector** the distance between apertures, L_0, the length of the collimator blades, L_1, the height from rotation axix to the slit centre, r_0, the rotation speed ω (in rpm), and the blade thickness t.

The local coordinate system has its Origo at the slit centre.

The component Selector produces equivalent results.

Velocity selector transmission

By rotating the selector you allow transmittance of neutrons rays with velocities around a nominal value, given by

$$V_0 = \omega L/\phi, \tag{6.11}$$

which means that the selector has turned the twist angle ϕ during the typical neutron flight time L/V_0. The actual twist angle is $\phi' = \omega t = \omega L/V$.

Neutrons having a velocity slightly different from V_0 will either be transmitted or absorbed depending on the exact position of the rotator blades when the neutron enters the selector. Assuming this position to be unknown and integrating over all possible positions (assuming zero thickness of blades), we arrive at a transmission factor

$$T = \begin{cases} 1 - (N/2\pi)|\phi - \omega L/V| & \text{if } (N/2\pi)|\phi - \omega L/V| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{6.12}$$

where N is the number of collimator blades.

A horisontal divergence changes the above formula because of the angular difference between the entry and exit points of the neutron. The resulting transmittance resembles the one above, only with V replaced by V_z and ϕ replaced by $(\phi + \psi)$, where ψ is the angular difference due to the divergence. An additional vertical divergence does not change this formula, but it may contribute to ψ. (We have here ignored the very small non-linearity of ψ along the neutron path in case of both vertical and horisontal divergence).

Adding the effect of a finite blade thickness, t, reduces the transmission by the overall factor

$$\left(1 - \frac{Nt}{2\pi r}\right), \tag{6.13}$$

where r is the distance from the rotation axis. We ignore the variation of r along the neutron path and use just the average value.

6.5. The `Selector` McStas Component

velocity selector (helical lamella type) such as V_selector component

Identification

- **Author:** Peter Link, Andreas Ostermann
- **Origin:** Uni. Gottingen (Germany)
- **Date:** MARCH 1999

Description

Velocity selector consisting of rotating Soller-like blades defining a helically twisted passage. Geometry is defined by two identical apertures at 12 o'clock position, The origin is at the ENTRANCE of the selector.

Example: Selector(xmin=-0.015, xmax=0.015, ymin=-0.025, ymax=0.025, length=0.25,

```
nslit=72,d=0.0004, radius=0.12, alpha=48.298, nu=500)
```

These are values for the D11@ILL Dornier 'Dolores' Velocity Selector (NVS 023)

%VALIDATION Jun 2005: extensive external test, one minor problem Jan 2006: problem solved (for McStas-1.9.1) Validated by: K. Lieutenant

%BUGS for transmission calculation, each neutron is supposed to be in the guide centre

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xmin	m	Lower x bound of entry aperture	-0.015
xmax	m	Upper x bound of entry aperture	0.015
ymin	m	Lower y bound of entry aperture	-0.025
ymax	m	Upper y bound of entry aperture	0.025
length	m	rotor length	0.25
xwidth	m	Width of entry. Overrides xmin,xmax.	0
yheight	m	Height of entry. Overrides ymin,ymax.	0
nslit	1	number of absorbing subdivinding spokes/lamella	72
d	m	width of spokes at beam-center	0.0004
radius	m	radius of beam-center	0.12
alpha	deg	angle of torsion	48.298
nu	Hz	frequency of rotation, which is ideally 3956*alpha*DEG2RAD/2/PI/lambda/length	500

Links

- Component source code found in file `Selector.comp`.
- See also Additional notes March 1999 and Jan 2000.

another approach to describe a rotating velocity selector

The component **Selector** describes the same kind of rotating velocity selector as **V_selector** - compare description there - but it uses different parameters and a different algorithm:

The position of the apertures relative to the z-axis (usually the beam centre) is defined by the four parameters $xmin, xmax, ymin, ymax$. Entry and exit apertures are always identical and situated directly before and behind the rotor. There are num blades of thickness $width$ twisted by the angle α (in degrees) on a length len. The selector rotates with a speed feq (in rotation per second); its axle is in a distance $radius$ below the z-axis.

First the neutron is propagated to the entrance window. The loss of neutrons hitting the thin side of the blades is taken into account by multiplying the neutron weight by a factor

$$p(r) = \theta_i(r)/\theta_o \tag{6.14}$$

$$\theta_o = 360^o/num \tag{6.15}$$

θ_i is the opening between two blades for the distance r between the neutron position (at the entrance) and the selector axle. The difference between θ_o and θ_i is determined by the blade thickness. The neutron is now propagated to the exit window. If it is outside the regarded channel (between the two actual blades), it is lost; otherwise it remains in the exit plane.

WARNING - Differences between **Selector** and **V_selector**:

- **Selector** has a different coordinate system than **V_selector**; in **Selector** the origin lies in the entrance plane of the selector.
- The blades are twisted to the other side, i.e. to the left above the axle in **Selector**.
- Speed of rotation is given in rotation per second, not in rotations per minute as in **V_selector**.

7. Polarising optics and magnetic field components

This chapter documents the components that read and/or write the neutron spin state (s_x, s_y, s_z): polarisation setting/analysis, polarising monochromators/mirrors/guides, and magnetic-field regions implementing spin precession. The underlying formalism – the polarisation vector, the master polarised-scattering equations, the F_N/F_M treatment of polarising monochromators and guides, the spin-precession algorithm, and the available magnetic field functions – is described in full in the Polarisation chapter (A); this chapter focuses on the component parameter tables themselves. See also section 8.1 above for **Monochromator_pol**.

7.1. Setting and analysing polarisation

7.2. The `Set_pol` McStas Component

(Unphysical) way of setting the polarization.

Identification

- **Author:** Peter Christiansen
- **Origin:** Risoe
- **Date:** August 2006

Description

This component has no physical size (like Arm - also drawn that way), but is used to set the polarisation in one of four ways: 1) (randomOn=0, normalize=0) Hardcode the polarisation to the vector (px, py, pz) 2) (randomOn!=0, normalize!=0) Set the polarisation to a random vector on the unit sphere 3) (randomOn!=0, normalize=0) Set the polarisation to a radnom vector within the unit sphere 4) (randomOn=0, normalize!=0) Hardcode the polarisation to point in the direction of (px,py,pz) but with polarization=1. Example: Set_pol(px=0, py=-1, pz=0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
px	1	X-component of polarisation vector (can be negative)	0
py	1	Y-component of polarisation vector (can be negative)	0
pz	1	Z-component of polarisation vector (can be negative)	0
randomOn	1	Generate random values if randomOn!=0.	0
normalize	1	Normalize the polarization vector to unity length.	0

Links

- Component source code found in file `Set_pol.comp`.

7.3. The `PolAnalyser_ideal` McStas Component

(Unphysical) ideal analyzer.

Identification

- **Author:** Erik Knudsen
- **Origin:** Risoe
- **Date:** June 2010

Description

This is an ideal model of a polarization analyser device. Given a polarization vector it "lets through" a scaled neutron weight. An imperfect analyzer could be modelled by an analysis vector M with |M|<1.

Example: PolAnalyser_ideal(mx=0, my=1, mz=0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
mx	1	X-component of polarisation analysis vector -1<mx<1	0
my	1	Y-component of polarisation analysis vector -1<my<1	0

Name	Unit	Description	Default
mz	1	Z-component of polarisation analysis vector -1<mz<1	0

Links

- Component source code found in file `PolAnalyser_ideal.comp`.

7.4. The `Pol_SF_ideal` McStas Component

Pol_SF_ideal component

Identification

- **Author:**
- **Origin:**
- **Date:**

Description

This component simply mirrors the polarization vector of the neutron ray in the plane through (0,0,0) with normal nx,ny,nz. The flipper is surrounded by a perfectly absorbing box. Neutron rays not hitting the box are left untouched.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nx		x-component of the normal vector of the flipping plane.	0
ny		y-component of the normal vector of the flipping plane.	1
nz		z-component of the normal vector of the flipping plane.	0
xwidth	m	width of the spin flipper.	0.1
yheight	m	height of the spin flipper.	0.1
zdepth	m	length of the spin flipper.	0.1

Links

- Component source code found in file `Pol_SF_ideal.comp`.

7.5. Polarising mirrors and guides

7.6. The `Pol_mirror` McStas Component

Polarising mirror.

Identification

- **Author:** Peter Christiansen
- **Origin:** RISOE
- **Date:** July 2006

Description

This component models a rectangular infinitely thin mirror. For an unrotated component, the mirror surface lies in the Y-Z plane (ie. parallel to the beam). It relies on similar physics as the Monochromator_pol. The reflectivity function (see e.g. share/ref-lib for examples) and parameters are passed to this component to give a bigger freedom. The up direction is hardcoded to be along the y-axis (0, 1, 0) IMPORTANT: At present the component only works correctly for polarization along the up/down direction and for completely unpolarized beams, i.e. sx=sy=sz=0 for all rays.

For now we assume: P(Transmit|Q) = 1 - P(Reflect|Q) i.e. NO ABSORPTION!

The component can both reflect and transmit neutrons with a respective proportion depending on the p_reflect parameter: p_reflect=-1 Reflect and transmit (proportions given from reflectivity) [default] p_reflect=1 Only handle reflected events p_reflect=0 Only handle transmitted events (reduce weight) p_reflect=0-1 Both transmit and reflect with fixed statistics proportions

The parameters can either be double pointer initializations (e.g. {R0, Qc, alpha, m, W}) or table names (e.g."supermirror_m2.rfl" AND useTables=1). NB! This might cause warnings by the compiler that can be ignored.

Examples: Reflection function parametrization Pol_mirror(zwidth = 0.40, yheight = 0.40, rUpFunc=StdReflecFunc, rUpPar={1.0, 0.0219, 6.07, 2.0, 0.003})

Table function Pol_mirror(zwidth = 0.40, yheight = 0.40, rUpFunc=TableReflecFunc, rUpPar="supermirror_m2.rfl", rDownFunc=TableReflecFunc, rDownPar="supermirror_m3.rfl", useTables=1)

See also the example instrument Test_Pol_Mirror (under tests).

GRAVITY: YES

%BUGS NO ABSORPTION

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
rUpPar	1	Parameters for rUpFunc	{0.99,0.0219,6.07,2.0,0.003}
rDownPar	1	Parameters for rDownFunc	{0.99,0.0219,6.07,2.0,0.003}
rUpData	str	Mirror Reflectivity data file for spin up	""
rDownData	str	Mirror Reflectivity data file for spin down	""
p_reflect	1	Proportion of reflected events. Use 0 to only get the transmitted beam, and 1 to get only the reflected beam. Use -1 to use the mirror reflectivity. This value is purely computational and is not related to the actual reflectivity	-1
zwidth	m	Width of the mirror	

Links

- Component source code found in file `Pol_mirror.comp`.

7.7. The `Pol_bender` McStas Component

Polarising bender.

Identification

- **Author:** Peter Christiansen
- **Origin:** RISOE
- **Date:** August 2006

Description

Based on Guide_curved written by Ross Stewart. Models a rectangular curved guide tube with entrance centered on the Z axis. The entrance lies in the X-Y plane. Draws a true depiction of the guide with multiple slits (but without spacers), and trajectories. It relies on similar physics as the Monochromator_pol. The reflec function and parameters are passed to this component to give a bigger freedom. The up direction is hardcoded to be along the y-axis (0, 1, 0)

The guide is asummed to have half a spacer on each side: slit1 slit2 slit3

```
|+     ++     ++     +|
|+     ++     ++     +|
<---------------------> xwidth
```

<——> xwidth/nslit (nslit=3) <> d

The reflection functions and parameters defaults as follows: Bot defaults to Top, Left defaults to Top, Right defaults to left. Down defaults to down and up defaults to up for all functions and Top(Up and Down) defaults to StdReflecFunc and {0.99,0.0219,6.07,2.0,0.003} which stands for {R0, Qc, alpha, m, W}.

Example: Pol_bender(xwidth = 0.08, yheight = 0.08, length = 1.0, radius= 10.0, nslit=5, d=0.0, endFlat=0, drawOption=2,

```
rTopUpPar={0.99, 0.0219, 6.07, 3.0, 0.003},
rTopDownPar={0.99, 0.0219, 6.07, 1.0, 0.003})
```

See also the example instruments Test_Pol_Bender and Test_Pol_Bender_Vs_Guide_Curved (under tests).

%BUGS This component has been against tested Guide_curved and found to give the same intensities. Gravity option has not been tested.

GRAVITY: YES (when gravity is along y-axis)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xwidth	m	Width at the guide entry	
yheight	m	Height at the guide entry	
length	m	length of guide along center	
radius	m	Radius of curvature of the guide (+:curve left/-:right)	
G	m/s^2	Gravitational constant	9.8
nslit	1	Number of slits	1
d	m	Width of spacers (subdividing absorbing walls)	0.0
debug	1	if debug > 0 print out some internal parameters	0
endFlat	1	If endflat>0 then entrance and exit planes are parallel.	0
rTopUpPar	1	Top mirror Parameters for spin up standard reflectivity function	{0.99,0.0219,6.07,2.0,(
rTopDownPar	1	Top mirror Parameters for spin down standard reflectivity function	{0.99,0.0219,6.07,2.0,(
rBotUpPar	1	Bottom mirror Parameters for spin up standard reflectivity function	{0.99,0.0219,6.07,2.0,(
rBotDownPar	1	Bottom mirror Parameters for spin down standard reflectivity function	{0.99,0.0219,6.07,2.0,(

Name	Unit	Description	Default
rLeftUpPar	1	Left mirror Parameters for spin up standard reflectivity function	{0.99,0.0219,6.07,2.0,0.003}
rLeftDownPar	1	Left mirror Parameters for spin down standard reflectivity function	{0.99,0.0219,6.07,2.0,0.003}
rRightUpPar	1	Right mirror Parameters for spin up standard reflectivity function	{0.99,0.0219,6.07,2.0,0.003}
rRightDownPar	1	Right mirror Parameters for spin down standard reflectivity function	{0.99,0.0219,6.07,2.0,0.003}
rTopUpData	1	Reflectivity file for top mirror, spin up	""
rTopDownData	1	Reflectivity file for top mirror, spin down	""
rBotUpData	1	Reflectivity file for bottom mirror, spin up	""
rBotDownData	1	Reflectivity file for bottom mirror, spin down	""
rLeftUpData	1	Reflectivity file for left mirror, spin up	""
rLeftDownData	1	Reflectivity file for left mirror, spin down	""
rRightUpData	1	Reflectivity file for right mirror, spin up	""
rRightDownData	1	Reflectivity file for right mirror, spin down	""

Links

- Component source code found in file `Pol_bender.comp`.

7.8. The `Pol_guide_mirror` McStas Component

Polarising guide with a supermirror along its diagonal.

Identification

- **Author:** Erik B Knudsen
- **Origin:** DTU Physics
- **Date:** July 2018

Description

Based on Pol_guide_vmirror by P. Christiansen. Models a rectangular guide with entrance centered on the Z axis and with one supermirror sitting on the diagonal inside. The entrance lies in the X-Y plane. Draws a true depiction of the guide with mirror and trajectories. The polarisation is handled similar to in Monochromator_pol. The reflec

functions are handled similar to Pol_mirror. The up direction is hardcoded to be along the y-axis (0, 1, 0)

Note that this component can also be used as a frame overlap-mirror if the up and down reflectivities are set equal. In this case the wall reflectivity (rPar) should probably be set to 0.

Reflectivity values can either come from datafiles or from sets of parameters to the standard analytic reflectivity function commonly used for neutron guides: R=R0; q<qc, R=(1-tanh((q-m qc)/W))(1-alpha(q-qc)). If a filename is specified for e.g. rData the datafile table overrides the analytic function.

GRAVITY: YES

%BUGS No absorption by mirror. The reflectivity parameters must be given as literal constants. Using variables will result in undefined behaviour.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
rUpPar	1	Mirror Parameters for spin up standard reflectivity function	{1.0, 0.0219, 4.07, 3.2, 0.003}
rDownPar	1	Mirror Parameters for spin down standard reflectivity function	{1.0, 0.0219, 4.07, 3.2, 0.003}
rPar	1	Guide Parameters for standard reflectivity function	{1.0, 0.0219, 4.07, 3.2, 0.003}
rData	str	Guide Reflectivity data file	""
rUpData	str	Mirror Reflectivity data file for spin up	""
rDownData	str	Mirror Reflectivity data file for spin down	""
xwidth	m	Width at the guide entry	
yheight	m	Height at the guide entry	
length	m	length of guide	

Links

- Component source code found in file `Pol_guide_mirror.comp`.

7.9. The `Pol_guide_vmirror` McStas Component

Polarising guide with nvs x 2 supermirrors sitting in v-shapes inside, upgraded from original single V-cavity.

Identification

- **Author:** Peter Willendrup
- **Origin:** DTU
- **Date:** June 2022

Description

Models a rectangular guide with entrance centered on the Z axis and with nvs x two supermirrors sitting in v-shapes inside. The entrance lies in the X-Y plane. Draws a true depiction of the guide with mirrors, and trajectories. The polarisation is handled similar to in Monochromator_pol. The reflec functions are handled similar to Pol_mirror. The up direction is hardcoded to be along the y-axis (0, 1, 0)

The reflectivity parameters can be 1. double pointer initializations with 5 parameters (e.g. {R0, Qc, alpha, m, W} AND inputType=0), or 2. double pointer initializations with 6 parameters (e.g. {R0, Qc, alpha, m, W, beta} AND inputType=0), or 3. table names (e.g."supermirror_m2.rfl" AND inputType=1), or 4. individually assigned values (e.g. rR0=1.0, rQc=0.0217, ... etc AND inputType=2). NB! This might cause warnings by the compiler that can be ignored. Note that the reflectivity functions that use with values and those that use tables are different.

GRAVITY: YES

%BUGS No absorption by mirror.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nvs	1	Number of V-cavities across width of guide	1
xwidth	m	Width at the guide entry	0.1
yheight	m	Height at the guide entry	0.1
length	m	length of guide	0.5
rR0	1	Cavity wall reflectivity below critical scattering vector, use with inputType = 2	1.0
rQc	$Å^{-1}$	Cavity wall reflectivity critical scattering vector, use natural Ni (m=1 definition) value of 0.0217, use with inputType = 2	0.0217
ralpha	Å	Cavity wall slope of reflectivity, use with inputType = 2	6.5

Name	Unit	Description	Default
rmSM	1	Cavity wall m-value of material. Zero means completely absorbing. Use with inputType = 2	2
rW	$Å^{-1}$	Cavity wall width of reflectivity cut-off, use with inputType = 2	0.00157
rbeta	$Å^2$	Cavity wall curvature of reflectivity, use with inputType = 2	80
rUpR0	1	Mirror spin up reflectivity below critical scattering vector, use with inputType = 2	1.0
rUpQc	$Å^{-1}$	Mirror spin up reflectivity critical scattering vector, use natural Ni (m=1 definition) value of 0.0217, use with inputType = 2	0.0217
rUpalpha	Å	Mirror spin up slope of reflectivity, use with inputType = 2	2.47
rUpmSM	1	Mirror spin up m-value of material. Zero means completely absorbing. Use with inputType = 2	4
rUpW	$Å^{-1}$	Mirror spin up width of reflectivity cut-off, use with inputType = 2	0.0014
rUpbeta	$Å^2$	Mirror spin up curvature of reflectivity, use with inputType = 2	0
rDownR0	1	Mirror spin down reflectivity below critical scattering vector, use with inputType = 2	1.0
rDownQc	$Å^{-1}$	Mirror spin down reflectivity critical scattering vector, use natural Ni (m=1 definition) value of 0.0217, use with inputType = 2	0.0217
rDownalpha	Å	Mirror spin down slope of reflectivity, use with inputType = 2	1
rDownmSM	1	Mirror spin down m-value of material. Zero means completely absorbing. Use with inputType = 2	0.65
rDownW	$Å^{-1}$	Mirror spin down width of reflectivity cut-off, use with inputType = 2	0.003
rDownbeta	$Å^2$	Mirror spin down curvature of reflectivity, use with inputType = 2	0
debug	1	if debug > 0 print out some internal runtime parameters	0

Name	Unit	Description	Default
allow_inside_start	1	Allow neutron to start inside cavity (no propagation to entry plane)	0
rPar	1	Cavity wall parameters array for rFunc, use with inputType = 0	{1.0, 0.0217, 6.5, 2, 0.00157, 80}
rUpPar	1	Mirror Parameters array for rUpFunc, use with inputType = 0	{1.0, 0.0217, 2.47, 4, 0.0014}
rDownPar	1	Mirror Parameters for rDownFunc, use with inputType = 0	{1.0, 0.0217, 1, 0.65, 0.003}
rParFile		Cavity wall parameters filename for rFunc, use with inputType = 1	""
rUpParFile		Mirror Parameters filename for rUpFunc, use with inputType = 1	""
rDownParFile		Mirror Parameters filename for rDown-Func, use with inputType = 1	""

Links

- Component source code found in file `Pol_guide_vmirror.comp`.

7.10. Magnetic field components

7.11. The `Pol_Bfield` McStas Component

Magnetic field component.

Identification

- **Author:** Erik B Knudsen, Peter Christiansen and Peter Willendrup
- **Origin:** RISOE
- **Date:** July 2011

Description

Region with a definable magnetic field.

The component is nestable if the concentric flag is set (the default). This means that it may have a construction like the following: // START MAGNETIC FIELD COMPONENT msf = Pol_Bfield(xwidth=0.08, yheight=0.08, zdepth=0.2, Bx=0, By=-0.678332e-4, Bz=0) AT (0, 0, 0) RELATIVE armMSF

// OTHER COMPONENTS INSIDE THE MAGNETIC FIELD CAN BE PLACED HERE
// STOP MAGNETIC FIELD COMPONENT msf_stop = Pol_simpleBfield_stop(magnet_comp_
AT (0, 0, 0) RELATIVE armMSF

Note that if you have objects within the magnetic field that extend outside it you may get wrong results, because propagation within the field will then possibly extend outside, e.g. when using a tabled field. The evaluated field will then use the nearest defined field point _also_ outside the defintion area. If these outside-points have a non-zero field precession will continue - even after the neutron has left the field region.

In between the two component instances the propagation routine PROP_DT also handles spin propagation. The current algorithm used for spin propagation is: SimpleNumMagnetPrecession in pol-lib.

Example: Pol_Bfield(xwidth=0.1, yheight=0.1, zdepth=0.2, Bx=0, By=1, Bz=0) Pol_Bfield(xwidth=0.1, yheight=0.1, zdepth=0.2, field_type=1)

Functions supplied by the system are (the number is the ID of the function to be given as the field_type parameter: 1. Constant magnetic field: Constant field (Bx,By,Bz) within the region 2. Rotating magnetic_field: Field is initially (0,By,0) but after a length of zdepth has rotated to (By,0,0) 3. Majorana magnetic_field: Field is initially (Bx,By,0) with By<<Bx, then linearly transforms to (-Bx,By,0) at z=zdepth. 4. MSF field: 5. RF field: A radio frequency field is modeled by an implcit use of a roating frame. 6. Gradient field: Similar to Majorana by without an x-component to the field. I.e. it (0,By,0) at z=0, and becomes (0,-By,0) AT z=zdepth.

If the concentric parameter is set to 1 the magnetic field is turned on by an instance of this component, and turned off by an instance of Pol_simpleBfield_stop. Anything in between is considered inside the field. If concentric is zero the field is considered a closed component - neutrons are propagated through the field region, and no other components are permitted inside the field.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
field_type		ID of function describing the magnetic field. 1:constant field, 2: rotating fiel, 3: majorana type field, 4: MSF, 5: RF-field, 6: gradient field.	0
xwidth	m	Width of opening.	0
yheight	m	Height of opening.	0
zdepth	m	Length of field.	0
radius	m	Radius of field if it is cylindrical or spherical.	0
Bx	T	Parameter used for x composant of field.	0

Name	Unit	Description	Default
By	T	Parameter used for y composant of field.	0
Bz	T	Parameter used for z composant of field.	0
concentric		Allow components and/or other fields inside field. Requires a subsequent Pol_simpleBfield_stop component.	1

Links

- Component source code found in file `Pol_Bfield.comp`.

7.12. The `Pol_Bfield_stop` McStas Component

Magnetic field component.

Identification

- **Author:** Erik B Knudsen, Peter Christiansen, and Peter Willendrup
- **Origin:** RISOE
- **Date:** August 2006

Description

End of magnetic field region defined by the latest preceeding Pol_Bfield component.

The component is concentric. It means that it requires a

// START MAGNETIC FIELD COMPONENT msf =

```
Pol_Bfield(xw=0.08, yh=0.08, length=0.2, Bx=0, By=-0.678332e-4, Bz=0)
```

AT (0, 0, 0) RELATIVE armMSF

// HERE CAN BE OTHER COMPONENTS INSIDE THE MAGNETIC FIELD

// STOP MAGNETIC FIELD COMPONENT msfCp = Pol_Bfield_stop() AT ("SOMEWHERE") RELATIVE armMSF

In between the two components the propagation routine PROP_DT also handles the spin propagation. The current algorithm used for spin propagation is: SimpleNumMagnetPrecession in pol-lib. and does not handle gravity.

GRAVITY: NO POLARISATION: YES

Example: Pol_Bfield_stop()

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
geometry	str	Name of an Object File Format (OFF) or PLY file for complex field-geometry.	""
yheight	m	Height of opening.	0
xwidth	m	Width of opening.	0
zdepth	m	Length of field.	0
radius	m	Radius of field if it is cylindrical or spherical.	0

Links

- Component source code found in file `Pol_Bfield_stop.comp`.

7.13. The `Pol_FieldBox` McStas Component

Box containing a constant magnetic field

Identification

- **Author:** Erik B Knudsen and P Willendrup
- **Origin:** Risoe
- **Date:** 2013

Description

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xwidth	m	Width of the box containing the field.	
yheight	m	Height of the box containing the field.	
zdepth	m	Depth of the box containing the field.	
Bx	T	Magnetic field strength along the x-axis.	0
By	T	Magnetic field strength along the y-axis.	1e-3
Bz	T	Magnetic field strength along the z-axis.	0

Links

- Component source code found in file `Pol_FieldBox.comp`.

7.14. The `Pol_constBfield` McStas Component

Constant magnetic field.

Identification

- **Author:** Peter Christiansen
- **Origin:** RISOE
- **Date:** July 2006

Description

Rectangular box with constant B field along y-axis (up). The component can be rotated to make either a guide field or a spin flipper. A neutron hitting outside the box opening or the box sides is absorbed.

Example: Pol_constBfield(xwidth=0.1, yheight=0.1, zdepth=0.2, fliplambda=5.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xwidth	m	Width of opening	
yheight	m	Height of opening	
zdepth	m	Length of field	
B	Gauss	Magnetic field along y-direction	0
fliplambda	Å	lambda for calculating B field	0
flipangle	deg	Angle flipped for lambda = fliplambda fliplambda and flipangle overrides B so a neutron with s= (1, 0, 0) and v = (0, 0, v(fliplambda)) will have s =(cos(flipangle), sin(flipangle), 0) after the section.	180

Links

- Component source code found in file `Pol_constBfield.comp`.

7.15. The `Pol_tabled_field` McStas Component

Magnetic field component.

Identification

- **Author:** Erik B Knudsen, Peter Christiansen and Peter Willendrup
- **Origin:** RISOE
- **Date:** July 2011

Description

Region with a tabled magnetic field read from file. The magnetic field is read from a text file where it is specified as a point cloud with N rows of 6 columns: x y z Bx By Bz the B field map is resampled with Stepx*Stepy*Stepz points. Use Stepx=Stepy=Stepz=0 to skip resampling and use the table as is. The regions itself may be either a 3D rectangular block, a cylinder with axis along y, or spherical. Interpolation is done between data-points.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xwidth	m	Width of opening.	0
yheight	m	Height of opening.	0
zdepth	m	Length of field.	0
radius	m	Radius of field if it is cylindrical or spherical.	0
filename	str	File where the magnetic field is tabulated.	"bfield.dat"
geometry	str	Name of an Object File Format (OFF) or PLY file for complex field-geometry.	NULL
interpol_method	str	Choice of interpolation method "kdtree" (default on CPU) / "regular" (default on GPU)	"default"

Links

- Component source code found in file `Pol_tabled_field.comp`.

8. Monochromators

In this class of components, we are concerned with elastic Bragg scattering from monochromators. **Monochromator_flat** models a flat thin mosaic crystal with a single scattering vector perpendicular to the surface. The component **Monochromator_curved** is physically similar, but models a singly or doubly bend monochromator crystal arrangement.

A much more general model of scattering from a single crystal is found in the component **Single_crystal**, which is presented under Samples, chapter 9.

8.1. The `Monochromator_flat` McStas Component

Flat Monochromator crystal with anisotropic mosaic.

Identification

- **Author:** Kristian Nielsen
- **Origin:** Risoe
- **Date:** 1999

Description

Flat, infinitely thin mosaic crystal, useful as a monochromator or analyzer. For an unrotated monochromator component, the crystal surface lies in the Y-Z plane (ie. parallel to the beam). The mosaic is anisotropic gaussian, with different FWHMs in the Y and Z directions. The scattering vector is perpendicular to the surface.

Example: Monochromator_flat(zmin=-0.1, zmax=0.1, ymin=-0.1, ymax=0.1, mosaich=30.0, mosaicv=30.0, r0=0.7, Q=1.8734)

Monochromator lattice parameter

```
PG       002 DM=3.355 AA (Highly Oriented Pyrolythic Graphite)
PG       004 DM=1.677 AA
```

Heusler 111 DM=3.362 Å (Cu2MnAl)

```
CoFe         DM=1.771 AA (Co0.92Fe0.08)
Ge       111 DM=3.266 AA
Ge       311 DM=1.714 AA
```

```
Ge       511 DM=1.089 AA
Ge       533 DM=0.863 AA
Si       111 DM=3.135 AA
Cu       111 DM=2.087 AA
Cu       002 DM=1.807 AA
Cu       220 DM=1.278 AA
Cu       111 DM=2.095 AA
```

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
zmin	m	Lower horizontal (z) bound of crystal	-0.05
zmax	m	Upper horizontal (z) bound of crystal	0.05
ymin	m	Lower vertical (y) bound of crystal	-0.05
ymax	m	Upper vertical (y) bound of crystal	0.05
zwidth	m	Width of crystal, instead of zmin and zmax	0
yheight	m	Height of crystal, instead of ymin and ymax	0
mosaich	arc minutes	Horizontal mosaic (in z direction) (FWHM)	30.0
mosaicv	arc minutes	Vertical mosaic (in y direction) (FWHM)	30.0
r0	1	Maximum reflectivity	0.7
Q	1/angstrom	Magnitude of scattering vector	1.8734
DM	Å	monochromator d-spacing, instead of Q = 2*pi/DM	0

Links

- Component source code found in file `Monochromator_flat.comp`.

An infinitely thin, flat mosaic crystal with a single scattering vector

This component simulates an infinitely thin single crystal with a single scattering vector, $Q_0 = 2\pi/d_m$, perpendicular to the surface. A typical use for this component is to simulate a simple monochromator or analyzer.

The monochromator dimensions are given by the length, z_{w}, and the height, y_{h}. As the parameter names indicate, the monochromator is placed in the $z-y$ plane of the local coordinate system. This definition is made to ensure that the physical monochromator angle (often denoted `A1`) will equal the McStas rotation angle of the Monochromator

component around the y-axis. R_0 is the maximal reflectivity and $\eta_{\rm h}$ and $\eta_{\rm v}$ are the horizontal and vertical mosaicities, respectively, see explanation below.

Monochromator physics and algorithm

The physical model used in **Monochromator_flat** is a rectangular piece of material composed of a large number of small micro-crystals. The orientation of the micro-crystals deviates from the nominal crystal orientation so that the probability of a given micro-crystal orientation is proportional to a Gaussian in the angle between the given and the nominal orientation. The width of the Gaussian is given by the mosaic spread, η, of the crystal (given in units of arc minutes). η is assumed to be large compared to the inherent Bragg width of the scattering vector (often a few arc seconds). (The mosaicity gives rise to a Gaussian reflectivity profile of width similar to - but not equal - the intrinsic mosaicity. In this component, and in real life, the mosaicity given is that of the reflectivity signal.)

As a further simplification, the crystal is assumed to be infinitely thin. This means that multiple scattering effects are not simulated. It also means that the total reflectivity, r_0 is used as a parameter for the model rather than the atomic scattering cross section, implying that the scattering efficiency does not vary with neutron wavelength. The variance of the lattice spacing ($\Delta d/d$) is assumed to be zero, so this component is not suitable for simulating backscattering instruments (use the component Single_crystal in section 9.4 for that).

When a neutron trajectory intersects the crystal, the first step in the computation is to determine the probability of scattering. This probability is then used in a Monte Carlo choice deciding whether to scatter or transmit the neutron. The physical scattering probability is the sum of the probabilities of first- second-, and higher-order scattering - up to the highest order possible for the given neutron wavelength. However, in most cases at most one order will have a significant scattering probability, and the computation thus considers only the order that best matches the neutron wavelength.

The scattering of neutrons from a crystal is governed by Bragg's law:

$$n\mathbf{Q}_0 = 2\mathbf{k}_i \sin\theta \tag{8.1}$$

The scattering order is specified by the integer n. We seek only one value of n, namely the one which makes $n\mathbf{Q}_0$ closest to the projection of $2\mathbf{k}_i$ onto $\mathbf{Q}_0$ (see figure 8.1).

Once n has been determined, the Bragg angle θ can be computed. The angle α is the amount one would need to turn the nominal scattering vector $\mathbf{Q}_0$ for the monochromator to be in Bragg scattering condition. We now use α to compute the probability of reflection from the mosaic crystal

$$p_{\rm reflect} = R_0 e^{-\alpha^2/2\eta^2}, \tag{8.2}$$

The probability $p_{\rm reflect}$ is used in a Monte Carlo choice to decide whether the neutron is transmitted or reflected.

In the case of reflection, the neutron will be scattered into the Debye-Scherrer cone, with the probability of each point on the cone being determined by the mosaic. The

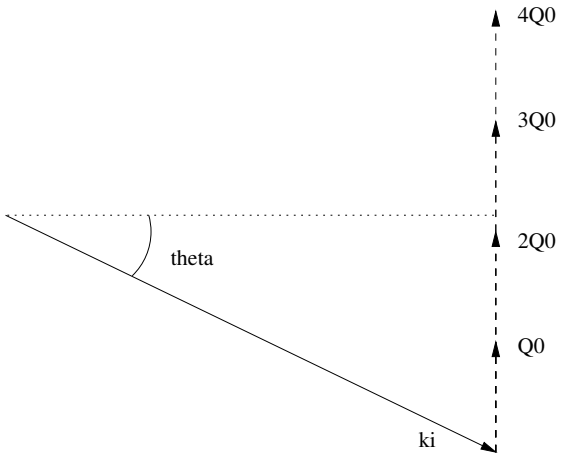


Figure 8.1.: Selection of the Bragg order ("2" in this case).

Debye-Scherrer cone can be described by the equation

$$\mathbf{k}_\mathrm{f} = \mathbf{k}_\mathrm{i}\cos 2\theta + \sin 2\theta(\mathbf{c}\cos\varphi + \mathbf{b}\sin\varphi), \qquad \varphi \in [-\pi;\pi], \tag{8.3}$$

where $\mathbf{b}$ is a vector perpendicular to $\mathbf{k}_\mathrm{i}$ and $\mathbf{Q}_0$, $\mathbf{c}$ is perpendicular to $\mathbf{k}_\mathrm{i}$ and $\mathbf{b}$, and both $\mathbf{b}$ and $\mathbf{c}$ have the same length as $\mathbf{k}_\mathrm{i}$ (see figure 8.2). When choosing φ (and thereby $\mathbf{k}_\mathrm{f}$), only a small part of the full $[-\pi;\pi]$ range will have appreciable scattering probability in non-backscattering configurations. The best statistics is thus obtained by sampling φ only from a suitably narrow range.

The (small) deviation angle α of the nominal scattering vector $n\mathbf{Q}_0$ corresponds to a Δq of

$$\Delta q \approx \alpha 2k\sin\theta. \tag{8.4}$$

The angle φ corresponds to a Δk_f (and hence Δq) of

$$\Delta q \approx \varphi k\sin(2\theta) \tag{8.5}$$

(see figure 8.2). Hence we may sample φ from a Gaussian with standard deviation

$$\alpha\frac{2k\sin\theta}{k\sin(2\theta)} = \alpha\frac{2k\sin\theta}{2k\sin\theta\cos\theta} = \frac{\alpha}{\cos\theta} \tag{8.6}$$

to get good statistics.

What remains is to determine the neutron weight. The distribution from which the scattering event is sampled is a Gaussian in φ of width $\frac{\alpha}{\cos\theta}$,

$$f_\mathrm{MC}(\varphi) = \frac{1}{\sqrt{2\pi}(\sigma/\cos\theta)}e^{-\varphi^2/2(\sigma/\cos\theta)^2} \tag{8.7}$$

In the physical model, the probability of the scattering event is proportional to a Gaussian in the angle between the nominal scattering vector $\mathbf{Q}_0$ and the actual scattering vector

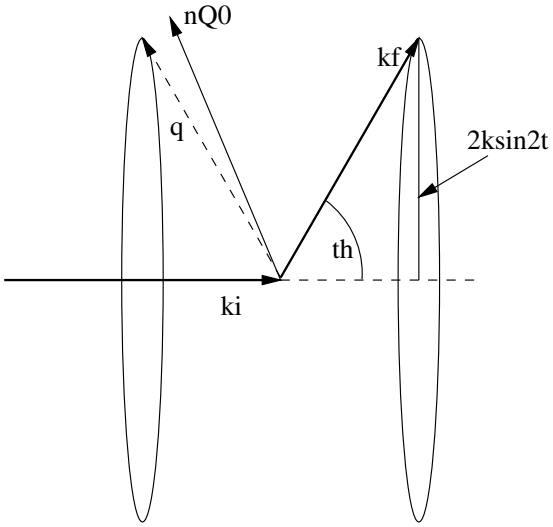


Figure 8.2.: Scattering into the part of the Debye-Scherrer cone covered by the mosaic.

q. The normalization condition is that the integral over all φ should be 1. Thus the probability of the scattering event in the physical model is

$$\Pi(\varphi) = e^{\frac{-d(\varphi)^2}{2\sigma^2}} / \int_{-\pi}^{\pi} e^{\frac{-d(\varphi)^2}{2\sigma^2}} d\varphi \tag{8.8}$$

where $d(\varphi)$ denotes the angle between the nominal scattering vector and the actual scattering vector corresponding to φ. According to equation (2.9), the weight adjustment π_j is then given by

$$\pi_j = \Pi(\varphi)/f_{\mathrm{MC}}(\varphi). \tag{8.9}$$

In the implementation, the integral in (8.8) is computed using a 15-order Gaussian quadrature formula, with the integral restricted to an interval of width $5\sigma/\cos\theta$ for the same reasons discussed above on the sampling of φ.

8.2. The `Monochromator_curved` McStas Component

Double bent multiple crystal slabs with anisotropic gaussian mosaic.

Identification

- **Author:** Emmanuel Farhi, Kim, Lefmann, Peter Link
- **Origin:** ILL
- **Date:** Aug. 24th 2001

Description

Double bent infinitely thin mosaic crystal, useful as a monochromator or analyzer. which uses a small-mosaicity approximation and taking into account higher order scattering. The mosaic is anisotropic gaussian, with different FWHMs in the Y and Z directions. The scattering vector is perpendicular to the surface. For an unrotated monochromator component, the crystal plane lies in the y-z plane (ie. parallel to the beam). The component works in reflection, but also transmits the non-diffracted beam. Reflectivity and transmission files may be used. The slabs are positioned in the vertical plane (not on a cylinder/sphere), and are rotated according to the curvature radius. When curvatures are set to 0, the monochromator is flat. The curvatures approximation for parallel beam focusing to distance L, with monochromator rotation angle A1 are: RV = 2*L*sin(DEG2RAD*A1); RH = 2*L/sin(DEG2RAD*A1);

When you rotate the component by A1 = asin(Q/2/Ki)*RAD2DEG, do not forget to rotate the following components by A2=2*A1 (for 1st order) !

Example: Monochromator_curved(zwidth=0.01, yheight=0.01, gap=0.0005, NH=11, NV=11, mosaich=30.0, mosaicv=30.0, r0=0.7, Q=1.8734)

Monochromator lattice parameter

```
PG       002 DM=3.355 AA (Highly Oriented Pyrolythic Graphite)
PG       004 DM=1.677 AA
```

Heusler 111 DM=3.362 Å (Cu2MnAl)

```
CoFe         DM=1.771 AA (Co0.92Fe0.08)
Ge       111 DM=3.266 AA
Ge       311 DM=1.714 AA
Ge       511 DM=1.089 AA
Ge       533 DM=0.863 AA
Si       111 DM=3.135 AA
Cu       111 DM=2.087 AA
Cu       002 DM=1.807 AA
Cu       220 DM=1.278 AA
Cu       111 DM=2.095 AA
```

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
reflect	str	reflectivity file name of text file as 2 columns [k, R]	"NULL"
transmit	str	transmission file name of text file as 2 columns [k, T]	"NULL"

Name	Unit	Description	Default
zwidth	m	horizontal width of an individual slab	0.01
yheight	m	vertical height of an individual slab	0.01
gap	m	typical gap between adjacent slabs	0.0005
NH	int	number of slabs horizontal	11
NV	int	number of slabs vertical	11
mosaich	arc minutes	Horisontal mosaic FWHM	30.0
mosaicv	arc minutes	Vertical mosaic FWHM	30.0
r0	1	Maximum reflectivity. O unactivates component	0.7
t0	1	transmission efficiency	1.0
Q	$Å^{-1}$	Scattering vector	1.8734
RV	m	radius of vertical focussing. flat for 0	0
RH	m	radius of horizontal focussing. flat for 0	0
DM	Å	monochromator d-spacing instead of Q=2*pi/DM	0
mosaic	arc minutes	sets mosaich=mosaicv	0
width	m	total width of monochromator, along Z	0
height	m	total height of monochromator, along Y	0
verbose	0/1	verbosity level	0
order	1	specify the diffraction order, 1 is usually prefered. Use 0 for all	0

Links

- Component source code found in file `Monochromator_curved.comp`.
- Additional note from Peter Link.
- Obsolete Mosaic_anisotropic by Kristian Nielsen
- Contributed Monochromator_2foc by Peter Link

A curved mosaic crystal with a single scattering vector

This component simulates an array of infinitely thin single crystals with a single scattering vector perpendicular to the surface and a mosaic spread. This component is used to simulate a singly or doubly curved monochromator or analyzer in reflecting geometry.

The component uses rectangular pieces of monochromator material as described in **Monochromator_curved**. The scattering vector is named Q, and as described in **Monochromator_flat**, multiples of Q will be applied. Other important parameters

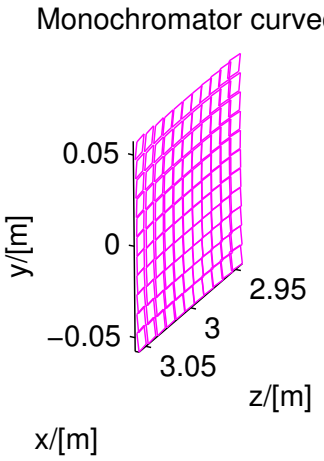


Figure 8.3.: A curved monochromator

are the piece height and width, y_{h} and z_{w}, respectively, the horizontal and vertical mosaicities, η_{h} and η_{v}, respectively. If just one mosaicity, η, is specified, this the same for both directions.

The number of pieces vertically and horizontally are called n_{v} and n_{h}, respectively, and the vertical and horizontal radii of curvature are named r_{v} and r_{h}, respectively. All single crystals are positioned in the same vertical plane, but tilted accordingly to the curvature radius.

The constant monochromator reflectivity, R_0 can be replaced by a file of tabulated reflectivities *reflect* (`*.rfl` in `MCSTAS/data`). In the same sense, the transmission can be modeled by a tabulated file *transmit* (for non-reflected neutrons, `*.trm` in `MCSTAS/data`). The most useful of these files for Monochromator_curved are `HOPG.rlf` and `HOPG.trm`.

As for **Monochromator_flat**, the crystal is assumed to be infinitely thin, and the variation in lattice spacing, $(\overline{\Delta d/d})$, is assumed to be zero. Hence, this component is not suitable for simulating backscattering instruments or to investigate multiple scattering effects.

The theory and algorithm for scattering from the individual blades is described under **Monochromator_flat**.

8.3. Monochromator_pol: A polarizing monochromator

The **Monochromator_pol** component is the polarising analogue of **Monochromator_flat** above – a flat, thin mosaic crystal with spin-dependent (up/down) reflectivity. Its underlying physics (the nuclear/magnetic structure factor formalism, F_N/F_M, and how they relate to the up/down reflectivities) is described in the User Manual's Polarisation chapter, section on Polarising Monochromator and Guides (A.2.2), which also documents **Pol_mirror**, **Pol_bender**, **Pol_guide_mirror** and **Pol_guide_vmirror** – see also section 7 of this manual for the rest of the polarisation-aware component set.

8.4. The `Monochromator_pol` McStas Component

Flat polarizaing monochromator crystal.

Identification

- **Author:** Peter Christiansen
- **Origin:** RISOE
- **Date:** 2006

Description

Based on Monochromator_flat. Flat, infinitely thin mosaic crystal, useful as a monochromator or analyzer. For an unrotated monochromator component, the crystal surface lies in the Y-Z plane (ie. parallel to the beam). The mosaic and d-spread distributions are both Gaussian. Neutrons are just reflected (billard ball like). No correction is done for mosaicity of reflecting crystal. The crystal is assumed to be a ferromagnet with spin pointing up eta-tilde = (0, 1, 0) (along y-axis), so that the magnetic field is pointing opposite (0, -|B|, 0).

The polarisation is done by defining the reflectivity for spin up (Rup) and spin down (Rdown) (which can be negative, see now!) and based on this the nuclear and magnetic structure factors are calculated: FM = sign(Rup)*sqrt(|Rup|) + sign(Rdown)*sqrt(|Rdown|) FN = sign(Rup)*sqrt(|Rup|) - sign(Rdown)*sqrt(|Rdown|) and the physics is calculated as Pol in = (sx_in, sy_in, sz_in) Reflectivity R0 = FN*FN + 2*FN*FM*sy_in + FM*FM (= |Rup| + |Rdown| (for sy_in=0)) Pol out: sx = (FN*FN - FM*FM)*sx_in/R0; sy = ((FN*FN - FM*FM)*sy_in + 2*FN*FM + FM*FM*sy_in)/R0; sz = (FN*FN - FM*FM)*sz_in/R0;

These equations are taken from: Lovesey: "Theory of neutron scattering from condensed matter, Volume

```
2", Eq. 10.96 and Eq. 10.110
```

This component works with gravity (uses PROP_X0).

Example: Monochromator_pol(zwidth=0.2, yheight=0.2, mosaic=30.0, dspread=0.0025, Rup=1.0, Rdown=0.0, Q=1.8734)

Monochromator lattice parameter

```
PG       002 DM=3.355 AA (Highly Oriented Pyrolythic Graphite)
PG       004 DM=1.677 AA
```

Heusler 111 DM=3.362 Å (Cu2MnAl)

```
CoFe         DM=1.771 AA (Co0.92Fe0.08)
Ge       111 DM=3.266 AA
Ge       311 DM=1.714 AA
Ge       511 DM=1.089 AA
Ge       533 DM=0.863 AA
Si       111 DM=3.135 AA
Cu       111 DM=2.087 AA
Cu       002 DM=1.807 AA
Cu       220 DM=1.278 AA
Cu       111 DM=2.095 AA
```

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
zwidth	m	Width of crystal	0.1
yheight	m	Height of crystal	0.1
mosaic	arcmin	Mosaicity (FWHM)	30.0
dspread	1	Relative d-spread (FWHM)	0
Q	$Å^{-1}$	Magnitude of scattering vector	1.8734
DM	Å	monochromator d-spacing instead of Q = 2*pi/DM	0
pThreshold		if probability>pThreshold then accept and weight else random	0
Rup	1	Reflectivity of neutrons with polarization up	1
Rdown	1	Reflectivity of neutrons with polarization down	1
debug		if debug > 0, print out some info about the calculations	0

Links

- Component source code found in file `Monochromator_pol.comp`.

8.5. Single_crystal: Thick single crystal monochromator plate with multiple scattering

The **Single_crystal** component may be used to study more complex monochromators, including incoherent scattering, thickness and multiple scattering. Please refer to section 9.4.

8.6. Phase space transformer - moving monochromator

Eventhough there exist a few attempts to write dedicated phase space transformer components, there is an elegant way to put a monochromator into move, by mean of the EXTEND keyword. If you define a SPEED parameter for the instrument, the idea is to change the coordinate system before the monochromator, and restore it afterwards, as follow in the TRACE section:

```
1  DEFINE INSTRUMENT PST(SPEED=200, ...)
2  (...)
3  TRACE
4  (...)
5  COMPONENT Mono_PST_on=Arm()
6  AT ...
7  EXTEND %{
8    vx = vx + SPEED; // monochromator moves transversaly by SPPED m/s
9  %}
10
11 COMPONENT  Mono=Monochromator(...)
12 AT (0,0,0) RELATIVE PREVIOUS
13
14 COMPONENT Mono_PST_off=Arm
15 AT (0,0,0) RELATIVE PREVIOUS
16 EXTEND  %{
17   vz = vz - SPEED; // puts back neutron in static coordinate frame
18 %}
```

This solution does not contain acceleration, but is far enough for most studies, and it is very simple. In the latter example, the instance `Mono_PST_on` should itself be rotated to reflect according to a Bragg law.

9. Samples

This class of components models the sample of the experiment. This is by far the most challenging part of a neutron scattering instrument to model. However, for purpose of simulating instrument performance, details of the samples are rather unimportant, allowing for simple approximations. On the contrary, for full virtual experiments it is of importance to have realistic and detailed sample descriptions. McStas contains both simple and detailed samples.

We first consider incoherent scattering. The general-purpose **Incoherent** component performs incoherent scattering and absorption for an arbitrary shape; the older, simpler **V_sample** component that previously served this role has moved to the **obsolete** component category (see the Component Manual's Obsolete chapter) and is superseded by **Incoherent**.

An important component class is elastic Bragg scattering from an ideal powder. The component **PowderN** models a powder scatterer with reflections given in an input file. To scatter on a single Bragg peak, the **Powder1** component may be used. The component includes absorption, incoherent scattering, direct beam transmission and can assume *concentric* shape, i.e. can be used for modelling sample enviroments.

Next type is Bragg scattering from single crystals. The simplest single crystals are in fact the monochromator components like **Monochromator_flat**, presented in section 8.1. The monochromators are models of a thin mosaic crystal with a single scattering vector perpendicular to the surface. Much more advanced, the component **Single_crystal** is a general single crystal sample (with multiple scattering) that allows the input of an arbitrary unit cell and a list of structure factors, read from a LAZY / Crystallographica file. This component also allows anisotropic mosaicity and $\Delta d/d$ lattice space variation.

Isotropic small-angle scattering is simulated in **Sans_Spheres**, which models scattering from a collection of hard spheres (dilute colloids).

Inelastic scattering from a dispersion is exemplified by the component **Phonon_simple**, which models scattering from a single acoustic phonon branch. Further dispersion samples (magnon scattering in **Magnon_bcc**, and many more in the `contrib` category), the magnetic single crystal **Single_magnetic_crystal**, and **NCrystal_sample**, which uses the external NCrystal library, are described at the end of this chapter and in the Contributed Components chapter.

For a more general sample model, the **Isotropic_Sqw** component is able to simulate all kinds of isotropic materials: Liquids, glasses, polymers, powders, etc, with $S(q,\omega)$ table specified by an input file. Physical processes include coherent/incoherent scattering, both elastic and inelastic, with absorption and multiple scattering. Moreover, this component may be used concentrically, to model a sample environment. Thus it may handle most samples except single crystals.

Sample	Coherent		Incoherent			
Process	Elastic	Inelastic	Elastic	Inelastic	Absorption	Multi. Scatt.
Phonon_simple		X			1	
Isotropic_Sqw	X	X	X	X	2	X
Powder1	1 line		X		1	
PowderN	N lines		X		1	
Sans_spheres	colloid				1	
Single_crystal	X		X		2	X
V_sample			X	QE broad.	1	
Tunneling_sample		X	X	QE broad.	1	

Table 9.1.: Processes implemented in sample components. Absorption: 1=single only, 2=with secondary

9.0.1. Neutron scattering notation

In sample components, we use the notation common for neutron scattering, where the wave vector transfer is denoted the *scattering vector*

$$\mathbf{q} \equiv \mathbf{k}_\mathrm{i} - \mathbf{k}_\mathrm{f}. \tag{9.1}$$

In analygo, the *energy transfer* is given by

$$\hbar\omega \equiv E_\mathrm{i} - E_\mathrm{f} = \frac{\hbar^2}{2m_\mathrm{n}} \left(k_\mathrm{i}^2 - k_\mathrm{f}^2\right). \tag{9.2}$$

9.0.2. Weight transformation in samples; focusing

Within many samples, the incident beam is attenuated by scattering and absorption, so that the illumination varies considerably throughout the sample. For single crystals, this phenomenon is known as *secondary extinction* [Bac75], but the effect is important for all samples. In analytical treatments, attenuation is difficult to deal with, and is thus often ignored, making a *thin sample approximation*. In Monte Carlo simulations, the beam attenuation is easily taken care of, as will be shown below. In the description, we ignore multiple scattering, which is however implemented in some sample components.

The sample has an absorption cross section per unit cell of σ_c^a and a scattering cross section per unit cell of σ_c^s. The neutron path length in the sample before the scattering event is denoted by l_1, and the path length within the sample after the scattering is denoted by l_2, see figure 9.1. We then define the inverse penetration lengths as $\mu^s = \sigma_c^s/V_c$ and $\mu^a = \sigma_c^a/V_c$, where V_c is the volume of a unit cell. Physically, the attenuation along this path follows

$$f_\mathrm{att}(l) = \exp(-l(\mu^s + \mu^a)), \tag{9.3}$$

where the normalization $f_\mathrm{att}(0) = 1$.

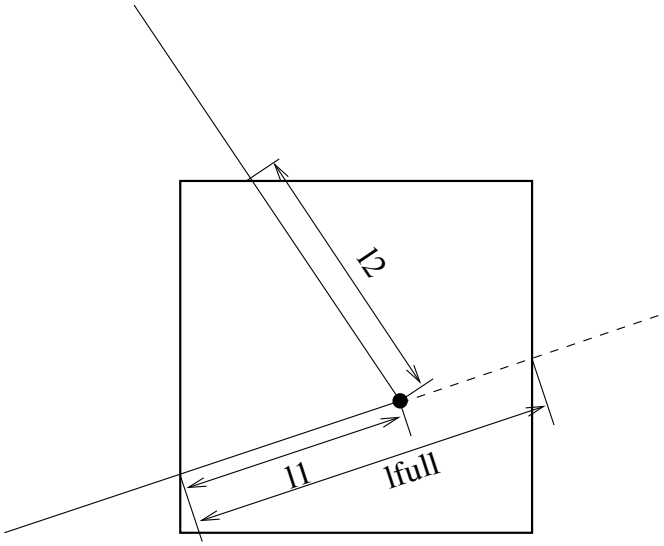


Figure 9.1.: The geometry of a scattering event within a powder sample.

The probability for a given neutron ray to be scattered from within the interval $[l_1; l_1 + dl]$ will be

$$P(l_1)dl = \mu^s f_{\text{att}}(l_1)dl, \tag{9.4}$$

while the probability for a neutron to be scattered from within this interval into the solid angle Ω *and* not being scattered further or absorbed on the way out of the sample is

$$P(l_1, \Omega)dld\Omega = \mu^s f_{\text{att}}(l_1) f_{\text{att}}(l_2)\gamma(\Omega)d\Omega dl, \tag{9.5}$$

where $\gamma(\Omega)$ is the directional distribution of the scattered neutrons, and l_2 is determined by Monte Carlo chocies of l_1, Ω, and from the sample geometry, see e.g. figure 9.1.

In our Monte-Carlo simulations, we may choose the scattering parameters by making a Monte-Carlo choice of l_1 and Ω from a distribution different from $P(l_1, \Omega)$. By doing this, we must adjust π_i according to the probability transformation rule (2.9). If we *e.g.* choose the scattering depth, l_1, from a flat distribution in $[0; l_{\text{full}}]$, and choose the directional dependence from $g(\Omega)$, we have a Monte Carlo probability

$$f(l_1, \Omega) = g(\Omega)/l_{\text{full}}, \tag{9.6}$$

l_{full} is here the path length through the sample as taken by a non-scattered neutron (although we here assume that all simulated neutrons are being scattered). According to (2.9), the neutron weight factor is now adjusted by the amount

$$\pi_i(l_1, \Omega) = \mu^s l_{\text{full}} \exp\left[-(l_1 + l_2)(\mu^a + \mu^s)\right] \frac{\gamma(\Omega)}{g(\Omega)}. \tag{9.7}$$

In analogy with the source components, it is possible to define "interesting" directions for the scattering. One will then try to focus the scattered neutrons, choosing a $g(\Omega)$,

which peaks around these directions. To do this, one uses (9.7), where the fraction $\gamma(\Omega)/g(\Omega)$ corrects for the focusing. One must choose a proper distribution so that $g(\Omega) > 0$ in every interesting direction. If this is not the case, the Monte Carlo simulation gives incorrect results. All samples have been constructed with a focusing and a non-focusing option.

9.0.3. Small-angle scattering (SANS) and other sample models

Beyond the simple isotropic small-angle model **Sans_spheres** (scattering from a dilute collection of hard spheres), McStas provides two further, much richer routes to SANS sample modelling:

- native McStas SANS models contributed in the `contrib` category (`SANSSpheres`, `SANSCylinders`, `SANSNanodiscs`, `SANSPDB`, and many others – see the Component Manual's Contributed Components chapter), several with polydisperse and multiple-scattering variants;
- the `sasmodels` category (see the Component Manual's dedicated chapter), exposing the same, actively maintained SANS/USANS form-factor library used for data fitting in SasView, as individual `SasView_*` components (the older, generic `SasView_model` component is obsolete).

In general, all samples are assumed to be homogeneous. There is potential in developing inhomogeneous samples, e.g. with a spatially varying lattice constant (relevant for stress/strain scanners) or inhomogeneous absorption (relevant for tomography), and reflectometry sample models beyond `Multilayer_Sample` (contrib). Polarization effects are handled by a dedicated set of polarisation-aware optics and field components rather than the samples described in this chapter – see the Polarisation chapter, A.

9.1. The `Incoherent` McStas Component

Incoherent sample (such as Vanadium) sample, with quasielastic component OR or global energy transfer.

Identification

- **Author:** Kim Lefmann and Kristian Nielsen
- **Origin:** Risoe
- **Date:** 15.4.98

Description

A Double-cylinder shaped incoherent scatterer (like Vanadium) with both elastic and quasielastic (Lorentzian) components. No multiple scattering (but approximation available). Absorption included. **Sample focusing:** The area to scatter to is a disk of radius

'focus_r' situated at the target. This target area may also be rectangular if specified focus_xw and focus_yh or focus_aw and focus_ah, respectively in meters and degrees. The target itself is either situated according to given coordinates (x,y,z), or defined with the relative target_index of the component to focus to (next is +1). This target position will be set to its AT position. When targeting to centered components, such as spheres or cylinders, define an Arm component where to focus to. **Sample shape:** Sample shape may be a cylinder, a sphere, a box or any other shape

```
box/plate:       xwidth x yheight x zdepth (thickness=0)
```

hollow box/plate:xwidth x yheight x zdepth and thickness>0

```
cylinder:        radius x yheight (thickness=0)
```

hollow cylinder: radius x yheight and thickness>0

```
sphere:          radius (yheight=0 thickness=0)
hollow sphere:   radius and thickness>0 (yheight=0)
any shape:       geometry=OFF, PLY or STL file
```

The complex geometry option handles any closed non-convex polyhedra. It computes the intersection points of the neutron ray with the object transparently, so that it can be used like a regular sample object. It supports the OFF (incl. NOFF, COFF), PLY and STL file formats. A mesh must be a single, closed object: the neutron path through it is taken from the first entry and exit only, so a second part of the mesh further along the path (disjoint parts, or a concave shape the path re-enters) is not seen. Such files may be generated from XYZ data using: qhull < coordinates.xyz Qx Qv Tv o > geomview.off or powercrust coordinates.xyz and viewed with geomview or java -jar jroff.jar (see below). The default size of the object depends of the OFF file data, but its bounding box may be resized using xwidth,yheight and zdepth.

Example: Incoherent(radius=0.05,focus_r=0.035, pack=1, target_index=1) Incoherent(geometry="socket.off",focus_r=0.035, pack=1, target_index=1)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
geometry	str	Name of an OFF, PLY or STL file for complex geometry. An OFF file may be generated from XYZ coordinates using qhull/powercrust	0
radius	m	Outer radius of sample in (x,z) plane	0

Name	Unit	Description	Default
xwidth	m	Horiz. dimension of sample (bounding box if off file), as a width	0
yheight	m	Vert. dimension of sample (bounding box if off file), as a height. A sphere shape is used when 0 and radius is set	0
zdepth	m	Depth of sample (bounding box if off file)	0
thickness	m	Thickness of hollow sample	0
target_x	m	position of target to focus at	0
target_y	m	position of target to focus at	0
target_z	m	position of target to focus at	0
focus_r	m	Radius of disk containing target. Use 0 for full space	0
focus_xw	m	horiz. dimension of a rectangular area	0
focus_yh	m	vert. dimension of a rectangular area	0
focus_aw	deg	horiz. angular dimension of a rectangular area	0
focus_ah	deg	vert. angular dimension of a rectangular area	0
target_index	1	Relative index of component to focus at, e.g. next is +1	0
pack	1	Packing factor	1
p_interact	1	MC Probability for scattering the ray; otherwise transmit	1
f_QE	1	Fraction of quasielastic scattering (rest is elastic)	0
gamma	1	Lorentzian width of quasielastic broadening (HWHM)	0
Etrans	meV	Global energy-transfer, for use in inelastic settings	0
deltaE	meV	Width in energy around Etrans, for use in inelastic settings	0
sigma_abs	barns	Absorption cross section pr. unit cell at 2200 m/s	5.08
sigma_inc	barns	Incoherent scattering cross section pr. unit cell	5.08
Vc	Å^3	Unit cell volume	13.827
concentric	1	Indicate that this component has a hollow geometry and may contain other components. It should then be duplicated after the inside part (only for box, cylinder, sphere)	0

Name	Unit	Description	Default
order	1	Limit multiple scattering up to given order 0 means all (default), 1 means single, 2 means double, ...	0

Links

- Component source code found in file `Incoherent.comp`.
- Cross sections for single elements
- Cross sections for compounds
- Web Elements
- <A HREF="http://neutron.risoe.dk/mcstas/components/tests/Incoherent/">Test
- results</A> (not up-to-date).
- The test/example instrument vanadium_example.instr.
- The test/example instrument QENS_test.instr.
- Geomview and Object File Format (OFF)
- Java version of Geomview (display only) jroff.jar
- qhull
- powercrust

An incoherent scatterer, the V-sample

A sample with incoherent scattering, e.g. vanadium, is frequently used for calibration purposes, as this gives an isotropic, elastically scattered beam.

The component **Incoherent** has *only* absorption and incoherent elastic scattering. For the sample geometry, we default use a hollow cylinder (which has the solid cylinder as a limiting case). The sample dimensions are: outer radius *radius*, height *yheight*, and (for a hollow cylinder) wall *thickness*, see figure 9.2.

Alternatively, the sample geometry can be made rectangular by specifying the width, *xwidth*, the height, *yheight*, and the depth, *zdepth*.

The incoherent and absorption cross sections for V are default for the component. For other choices, the parameters *sigma_ inc*, *sigma_ abs*, and the unit cell volume *Vc* should be specified. For a loosely packed sample, also the packing factor, *pack* can be specified (default value of 1).

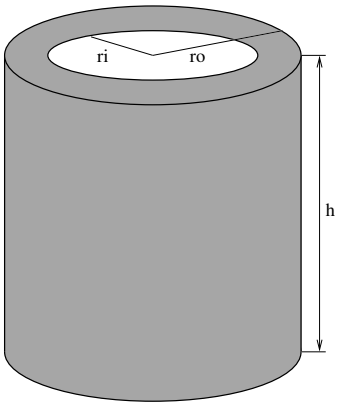


Figure 9.2.: The geometry of the hollow-cylinder vanadium sample.

9.1.1. Physics and algorithm

The incoherent scattering gives a uniform angular distribution of the scattered neutrons from each nucleus: $\gamma(\Omega) = 1/4\pi$. For the focusing we choose to have a uniform distribution on a target sphere of radius *focus_ r*, at the position (*target_ x*,*target_ y*,*target_ z*) in the local coordinate system. This gives an angular distribution (in a small angle approximation) of

$$g(\Omega) = \frac{1}{4\pi}\frac{x_t^2 + y_t^2 + z_t^2}{(\pi r_t^2)}. \tag{9.8}$$

The focusing can alternatively be performed on a rectangle with dimensions *focus_ xw*, *focus_ yh*, or uniformly in angular space (in a small-angle approximation), using *focus_ aw*, *focus_ ah*. The focusing location can be picked to be a downstream component by specifying
`target_index`.

When calculating the neutron path length within the cylinder, the kernel function `cylinder_intersect` is used twice, once for the outer radius and once for the inner radius.

Multiple scattering is not included in this component. To obtain intensities similar to real measured ones, we therefore do not take attenuation from scattering into account for the outgoing neutron ray.

9.1.2. Remark on functionality

When simulating a realistic incoherent hollow cylinder sample one finds that the resulting direction dependence of the scattered intensity is *not* isotropic. This is explained by the variation of attenuation with scattering angle. One test result is shown in the instrument example chapter of the McStas User Manual.

The `Samples_vanadium` and `Samples_incoherent` test/example instruments exist in the distribution for this component.

9.2. The `Tunneling_sample` McStas Component

A Double-cylinder shaped all-incoherent scatterer with elastic, quasielastic (Lorentzian), and tunneling (sharp) components.

Identification

- **Author:** Kim Lefmann
- **Origin:** Risoe
- **Date:** 10.05.07

Description

A Double-cylinder shaped all-incoherent scatterer with both elastic, quasielastic (Lorentzian), and tunneling (sharp) components. No multiple scattering. Absorbtion included. The shape of the sample may be a box with dimensions xwidth, yheight, zdepth. With thickness>0 the cylinder is hollow, with no lids: the cavity spans the full yheight, as a sample can without lids. The area to scatter to is a disk of radius 'focus_r' situated at the target. This target area may also be rectangular if specified focus_xw and focus_yh or focus_aw and focus_ah, respectively in meters and degrees. The target itself is either situated according to given coordinates (x,y,z), or defined with the relative target_index of the component to focus to (next is +1). This target position will be set to its AT position. When targeting to centered components, such as spheres or cylinders, define an Arm component where to focus to.

The outgoing polarization is calculated as for nuclear spin incoherence:

```
P' = 1/3*P-2/3P = -1/3P
```

As above multiple scattering is ignored .

Example: Tunneling_sample(thickness=0.001,radius=0.01,yheight=0.02,focus_r=0.035, target_index=1)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
thickness	m	Thickness of cylindrical sample in (x,z) plane	0
radius	m	Outer radius of sample in (x,z) plane	0.01
focus_r	m	Radius of disk containing target. Use 0 for full space	0

Name	Unit	Description	Default
p_interact	1	MC Probability for scattering the ray; otherwise transmit	1
f_QE	1	Fraction of quasielastic scattering	0
f_tun	1	Fraction of tunneling scattering (f_QE+f_tun $<$ 1)	0
gamma	meV	Lorentzian width of quasielastic broadening (HWHM)	0
E_tun	meV	Tunneling energy	0
target_x	m	X-position of target to focus at	0
target_y	m	Y-position of target to focus at	0
target_z	m	Z-position of target to focus at	0.235
focus_xw	m	horiz. dimension of a rectangular area	0
focus_yh	m	vert. dimension of a rectangular area	0
focus_aw	deg	horiz. angular dimension of a rectangular area	0
focus_ah	deg	vert. angular dimension of a rectangular area	0
xwidth	m	horiz. dimension of sample, as a width	0
yheight	m	vert. dimension of sample, as a height	0.05
zdepth	m	depth of sample	0
sigma_abs	barns	Absorbtion cross section pr. unit cell	5.08
sigma_inc	barns	Total incoherent scattering cross section pr. unit cell	4.935
Vc	Å^3	Unit cell volume	13.827
target_index	1	relative index of component to focus at, e.g. next is +1	0

Links

- Component source code found in file `Tunneling_sample.comp`.
- <A HREF="http://neutron.risoe.dk/mcstas/components/tests/v_sample/">Test
- results</A> (not up-to-date).

An incoherent inelastic scatterer

The component **Tunneling_sample** displays incoherent inelastic scattering as found in a number of systems, *e.g.* containing mobile hydrogen.

For the sample geometry, we default use a hollow cylinder (which has the solid cylinder as a limiting case). The sample dimensions are: Inner radius r_i, outer radius r_o, and height h. This geometry is the same as the default for **V_sample**, see figure 9.2.

As for **V_sample**, the sample geometry can be made rectangular by specifying the width, w_x, the height, h_y, and the thickness, t_z.

Also the focusing properties are the same as for **V_sample**. For the focusing is performed as a uniform distribution on a target sphere of radius r_{foc}, at the position $(x_{\mathrm{target}}, y_{\mathrm{target}}, z_{\mathrm{target}})$ in the local coordinate system. The focusing can alternatively be performed on a rectangle with dimensions w_{focus}, h_{focus}, or uniformly in angular space (in a small-angle approximation), using $w_{\mathrm{foc,\ angle}}$, $h_{\mathrm{foc,\ angle}}$. The focusing location can be picked to be a downstream component by specifying `target_index`.

The incoherent and absorption cross sections for V are default for the component. For other choices, the parameters σ_{inc}, σ_{abs}, and the unit cell volume V_0 should be specified. For a loosely packed sample, also the packing factor, f_{pack} can be specified (default value of 1).

The inelastic scattering takes place as a quasielastic (Lorentzian) component, which is chosen with probability f_{QE}. The broadening of the signal is given by Γ (HWHM). In addition, a tunneling signal is present with a probability of f_{tun} and a tunneling energy of $\pm E_{\mathrm{tun}}$. The tunneling peaks are weighted by the usual factor $k_{\mathrm{f}}/k_{\mathrm{i}}$.

The total scattering cross section is given by

$$\begin{aligned}\frac{d^2\sigma}{d\Omega dE_{\mathrm{f}}}(q,\omega) = \frac{\sigma_{\mathrm{inc}}}{4\pi} \times \{&(1 - f_{\mathrm{QE}} - f_{\mathrm{inel}})\delta(\hbar\omega) \\ + \quad & f_{\mathrm{QE}} \frac{\Gamma}{(\hbar\omega)^2 + \Gamma^2} + \frac{f_{\mathrm{inel}}}{2}\frac{k_{\mathrm{f}}}{f_{\mathrm{i}}}\left[\delta(\hbar\omega - E_{\mathrm{tun}}) + \delta(\hbar\omega + E_{\mathrm{tun}})\right]\Big\}\end{aligned} \tag{9.9}$$

The component takes care that $f_{\mathrm{QE}} + f_{\mathrm{tun}} \leq 1$, otherwise an error is returned.

The component accounts for absorption, but not multiple scattering. To obtain intensities similar to real measured ones, we therefore do not take attenuation from scattering into account for the outgoing neutron ray.

9.3. The PowderN McStas Component

General powder sample (N lines, single scattering, incoherent scattering)

Identification

- **Author:** P. Willendrup, L. Chapon, K. Lefmann, A.B.Abrahamsen, N.B.Christensen, E.M.Lauridsen.
- **Origin:** McStas release
- **Date:** 4.2.98

Description

General powder sample with many scattering vectors possibility for intrinsic line broadening incoherent elastic background ratio is specified by user No multiple scattering. No secondary extinction.

Based on Powder1/Powder2/Single_crystal. Geometry is a powder filled cylinder, sphere, box or any shape from an OFF file. Incoherent scattering is only provided here to account for a background. The efficient is highly improved when restricting the vertical scattering range on the Debye-Scherrer cone (with 'd_phi' and 'focus_flip'). The unit cell volume Vc may also be computed when giving the density, the atomic/molecular weight and the number of atoms per unit cell. A simple strain handling is available by mean of either a global Strain parameter, or a column with a strain value per Bragg reflection. The strain values are specified in ppm (1e-6). The Single_crystal component can also handle a powder mode, as well as an approximated texture.

Sample shape: Sample shape may be a cylinder, a sphere, a box or any other shape.

```
box/plate:        xwidth x yheight x zdepth (thickness=0)
```

hollow box/plate:xwidth x yheight x zdepth and thickness>0

```
cylinder:         radius x yheight (thickness=0)
```

hollow cylinder: radius x yheight and thickness>0 (no lids: the cavity spans the full yheight, as a sample can without lids)

```
sphere:           radius (yheight=0 thickness=0)
hollow sphere:    radius and thickness>0 (yheight=0)
any shape:        geometry=OFF_file
```

The complex geometry option handles any closed non-convex polyhedra. It computes the intersection points of the neutron ray with the object transparently, so that it can be used like a regular sample object. It supports the OFF (incl. NOFF, COFF), PLY and STL file formats. A mesh must be a single, closed object: the neutron path through it is

taken from the first entry and exit only, so a second part of the mesh further along the path (disjoint parts, or a concave shape the path re-enters) is not seen. Such files may be generated from XYZ data using: qhull < coordinates.xyz Qx Qv Tv o > geomview.off or powercrust coordinates.xyz and viewed with geomview or java -jar jroff.jar (see below). The default size of the object depends of the OFF file data, but its bounding box may be resized using xwidth,yheight and zdepth.

If you use this component and produce valuable scientific results, please cite authors with references bellow (in Links).

Example: PowderN(reflections = "c60.lau", d_phi = 15 , radius = 0.01, yheight = 0.05, Vc = 1076.89, sigma_abs = 0, delta_d_d=0, DW=1))

Powder definition file format Powder structure is specified with an ascii data file 'reflections'. The powder data are free-text column based files. The reflection list should be ordered by decreasing d-spacing values.

```
... d ... F2
```

Lines begining by '#' are read as comments (ignored) but they may contain the following keywords (in the header):

```
#Vc           <value of unit cell volume Vc [Angs^3]>
#sigma_abs    <value of Absorption cross section [barns]>
#sigma_inc    <value of Incoherent cross section [barns]>
```

#Debye_Waller <value of Debye-Waller factor DW>

```
#delta_d_d/d    <value of delta_d_d/d width for all lines>
```

These values are not read if entered as component parameters (Vc=...)

The signification of the columns in the numerical block may be set using the 'format' parameter, by defining signification of the columns as a vector of indexes in the order format={j,d,F2,DW,delta_d_d/d,1/2d,q,F,Strain}

Signification of the symbols is given below. Indices start at 1. Indices with zero means that the column are not present, so that: Crystallographica={ 4,5,7,0,0,0,0,0,0 }

```
Fullprof         ={ 4,0,8,0,0,5,0,0,0 }
Lazy             ={17,6,0,0,0,0,0,13,0}
```

At last, the format may be overridden by direct definition of the column indexes in the file itself by using the following keywords in the header (e.g. '#column_j 4'):

```
#column_j     <index of the multiplicity 'j' column>
#column_d     <index of the d-spacing 'd' column [Angs]>
#column_F2    <index of the squared str. factor '|F|^2' column [b]>
#column_F     <index of the structure factor norm '|F|' column>
#column_DW    <index of the Debye-Waller factor 'DW' column>
#column_Dd    <index of the relative line width delta_d_d/d broadening 'Dd' column>
```

#column_inv2d <index of the 1/2d=sin(theta)/lambda 'inv2d' column>

```
#column_q     <index of the scattering wavevector 'q' column [Angs-1]>
```

#column_strain <index of the strain line shift Delta/d [ppm]>

Last, CIF, FullProf and ShelX files can be read, and converted to F2(hkl) lists if 'cif2hkl' is installed. The CIF2HKL env variable can be used to point to a proper executable, else the McCode, then the system installed versions are used.

NCrystal materials The 'reflections' parameter can also be an NCrystal cfg-string (containing ".ncmat" or "::"), e.g. "stdlib::Al_sg225.ncmat" or "mycrystal.ncmat". The reflections, unit cell and cross sections are then taken from NCrystal. Multiphase materials are supported, e.g. "phases<0.3*stdlib::Al_sg225.ncmat&0.7*stdlib::Cu_sg225.ncmat>" but oriented materials are not. Only Bragg diffraction, absorption and incoherent scattering are modelled. For the full NCrystal physics, use the NCrystal_sample component, or the NCrystal_process component with Union. **Concentricity**

PowderN assumes 'concentric' shape, i.e. can contain other components inside its optional inner hollow. Example, Sample in Al cryostat:

COMPONENT Cryo = PowderN(reflections="Al.laz", radius = 0.01, thickness = 0.001, concentric = 1, p_interact=0.1) AT (0,0,0) RELATIVE Somewhere

COMPONENT Sample = some_other_component(with geometry FULLY enclosed in the hollow) AT (0,0,0) RELATIVE Somewhere

COMPONENT Cryo2 = COPY(Cryo)(concentric = 0) AT (0,0,0) RELATIVE Somewhere

(The second instance of the cryostat component can also be written out completely using PowderN(...). In both cases, this second instance needs concentric = 0.) The concentric arrangment can not be used with OFF geometry specification.

This sample component can advantageously benefit from the SPLIT feature, e.g. SPLIT COMPONENT pow = PowderN(...)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
reflections	string	Input file for reflections (LAZ LAU CIF, FullProf, ShelX, NCrystal). Use only incoherent scattering if NULL or ""	"NULL"
geometry	str	Name of an OFF, PLY or STL file for complex geometry. An OFF file may be generated from XYZ coordinates using qhull/powercrust.	"NULL"
format	no quotes	Name of the format, or list of column indexes (see Description).	{0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
radius	m	Outer radius of sample in (x,z) plane	0

Name	Unit	Description	Default
yheight	m	Height of sample y direction	0
xwidth	m	Horiz. dimension of sample, as a width	0
zdepth	m	Depth of box sample	0
thickness	m	Thickness of hollow sample. Negative value extends the hollow volume outside of the box/cylinder.	0
pack	1	Packing factor.	1
Vc	Å^3	Volume of unit cell=nb atoms per cell/density of atoms.	0
sigma_abs	barns	Absorption cross section per unit cell at 2200 m/s. Use a negative value to inactivate it.	0
sigma_inc	barns	Incoherent cross section per unit cell. Use a negative value to inactivate it.	0
delta_d_d	0/1	Global relative delta_d_d/d broadening when the 'w' column is not available. Use 0 if ideal.	0
p_inc	1	Fraction of incoherently scattered neutron rays.	0.1
p_transmit	1	Fraction of transmitted (only attenuated) neutron rays.	0.1
DW	1	Global Debye-Waller factor when the 'DW' column is not available. Use 1 if included in F2	0
nb_atoms	1	Number of sub-unit per unit cell, that is ratio of sigma for chemical formula to sigma per unit cell	1
d_omega	deg	Horizontal focus range (only for incoherent scattering), 0 for no focusing.	0
d_phi	deg	Angle corresponding to the vertical angular range to focus to, e.g. detector height. 0 for no focusing.	0
tth_sign	1	Sign of the scattering angle. If 0, the sign is chosen randomly (left and right). ONLY functional in combination with d_phi and ONLY applies to bragg lines.	0
p_interact	1	Fraction of events interacting coherently with sample.	0.8

Name	Unit	Description	Default
concentric	1	Indicate that this component has a hollow geometry and may contain other components. It should then be duplicated after the inside part (only for box, cylinder, sphere).	0
density	g/cm^3	Density of material. rho=density/weight/1e24*N_A.	0
weight	g/mol	Atomic/molecular weight of material.	0
barns	1	Flag to indicate if |F|^2 from 'reflections' is in barns or fm^2 (barns=1 for laz, barns=0 for lau type files).	1
Strain	ppm	Global relative delta_d_d/d shift when the 'Strain' column is not available. Use 0 if ideal.	0
focus_flip	1	Controls the sense of d_phi. If 0 d_phi is measured against the xz-plane. If !=0 d_phi is measured against zy-plane.	0

Links

- Component source code found in file `PowderN.comp`.
- "Validation of a realistic powder sample using data from DMC at PSI" Willendrup P, Filges U, Keller L, Farhi E, Lefmann K, Physica B-Cond Matt 385 (2006) 1032.
- See also: Powder1, Single_crystal
- See ICSD Inorganic Crystal Structure Database
- Cross sections for single elements
- Cross sections for compounds
- Web Elements
- Fullprof powder refinement
- Crystallographica software (free license)
- Geomview and Object File Format (OFF)
- Java version of Geomview (display only) jroff.jar
- qhull
- powercrust

A general powder sample

The powder diffraction component **PowderN** models a powder sample with background coming only from incoherent scattering and no multiple scattering. At the users choice, a given percentage of the incoming events may be transmitted (attenuated) to model the direct beam. The component can also assume *concentric* shape, i.e. be used for describing sample environment (cryostat, sample container etc.).

The description of the powder comes from a file in one of the standard output formats LAZY, FULLPROF, or CRYSTALLOGRAPHICA.

A usage example of this component can be found in the `Neutron site/Tutorial/templateDIFF` instrument from the `mcgui`.

9.3.1. Files formats: powder structures

Data files of type `lau` and `laz` in the McStas distribution data directory are self-documented in their header. A list of common powder definition files is available in Table 1.2 (page 24). They do not need any additional parameters to be used, as in the example:

```
PowderN(<geometry parameters>, filename="Al.laz")
```

Other column-based file formats may also be imported e.g. with parameters such as:

```
format=Crystallographica
format=Fullprof
format={1,2,3,4,0,0,0,0}
```

In the latter case, the indices define order of columns parameters multiplicity, lattice spacing, F^2, Debye-Waller factor and intrinsic line width.

The column signification may as well explicitely be set in the data file header using any of the lines:

```
#column_j      <index of the multiplicity 'j' column>
#column_d      <index of the d-spacing 'd' column>
#column_F2     <index of the squared str. factor '|F|^2' column [b]>
#column_F      <index of the structure factor norm '|F|' column>
#column_DW     <index of the Debye-Waller factor 'DW' column>
#column_Dd     <index of the relative line width Delta_d/d 'Dd' column>
#column_inv2d <index of the 1/2d=sin(theta)/lambda 'inv2d' column>
#column_q      <index of the scattering wavevector 'q' column>
```

Other component parameters may as well be specified in the data file header with lines e.g.:

```
#V_rho         <value of atom number density [at/Angs^3]>
#Vc            <value of unit cell volume Vc [Angs^3]>
#sigma_abs     <value of Absorption cross section [barns]>
#sigma_inc     <value of Incoherent cross section [barns]>
#Debye_Waller <value of Debye-Waller factor DW>
#Delta_d/d     <value of Delta_d/d width for all lines>
#density       <value of material density [g/cm^3]>
#weight        <value of material molar weight [g/mol]>
```

```
9  #nb_atoms      <value of number of atoms per unit cell>
```

Further details on file formats are available in the `mcdoc` page of the component.

9.3.2. Geometry, physical properties, concentricity

The sample has the shape of a solid cylinder, radius r and height h or a box-shaped sample of size *xwidth* x *yheight* x *zdepth*. At the users choice, an inner 'hollow' can be specified using the parameter *thickness*.

As the Isotropic_Sqw component 9.7, PowderN assumes *concentric* shape, i.e. can contain other components inside the inner hollow. To allow this, two almost identical copies of the PowderN components must be set up *around* the internal component(s), for example:

```
1 COMPONENT Cryo = PowderN(reflections="Al.laz", radius = 0.01, thickness =
      0.001,
2                          concentric = 1)
3 AT (0,0,0) RELATIVE Somewhere
4
5 COMPONENT Sample = some_other_component(with geometry FULLY enclosed in the
       hollow)
6 AT (0,0,0) RELATIVE Somewhere
7
8 COMPONENT Cryo2 = COPY(Cryo)(concentric = 0)
9 AT (0,0,0) RELATIVE Somewhere
```

As outlined, the first instance of PowderN *must* have `concentric = 1` and the instance *must* have `concentric = 0`. Furthermore, the component(s) inside the hollow *must* have a geometry which can be fully contained inside the hollow.

In addition to the coherent scattering specified in the `reflections` file, absorption- and incoherent cross sections can be given using the input parameters σ_c^a and σ_i^s.

The Bragg scattering from the powder, σ_c^s is calculated from the input file, with the parameters Q, $|F(Q)|^2$, and j for the scattering vector, structure factor, and multiplicity, respectively. The volume of the unit cell is denoted Vc, while the sample packing factor is f_{pack}.

Focusing is performed by only scattering into one angular interval, $d\phi$ of the Debye-Scherrer circle. The center of this interval is located at the point where the Debye-Scherrer circle intersects the half-plane defined by the initial velocity, $\mathbf{v}_\text{i}$, and a user-specified vector, $\mathbf{f}$.

9.3.3. Powder scattering

An ideal powder sample consists of many small crystallites, although each crystallite is sufficiently large not to cause measurable size broadening. The orientation of the crystallites is evenly distributed, and there is thus always a large number of crystallites oriented to fulfill the Bragg condition

$$n\lambda = 2d\sin\theta, \tag{9.10}$$

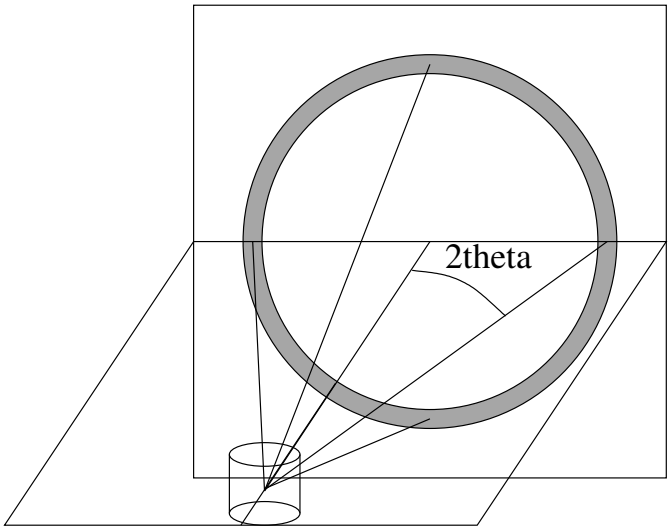


Figure 9.3.: The scattering geometry of a powder sample showing part of the Debye-Scherrer cone (solid lines) and the Debye-Scherrer circle (grey).

where n is the order of the scattering (an integer), λ is the neutron wavelength, d is the lattice spacing of the sample, and 2θ is the scattering angle, see figure 9.3. As all crystal orientations are realised in a powder sample, the neutrons are scattered within a *Debye-Scherrer cone* of opening angle 4θ [Bac75].

Equation (9.10) may be cast into the form

$$|\mathbf{Q}| = 2|\mathbf{k}| \sin\theta, \tag{9.11}$$

where $\mathbf{Q}$ is a vector of the reciprocal lattice, and $\mathbf{k}$ is the wave vector of the neutron. It is seen that only reciprocal vectors fulfilling $|\mathbf{Q}| < 2|\mathbf{k}|$ contribute to the scattering. For a complete treatment of the powder sample, one needs to take into account all these $\mathbf{Q}$-values, since each of them contribute to the attenuation.

The strength of the Bragg reflections is given by their structure factors

$$\left| \sum_j b_j \exp(\mathbf{R}_j \cdot \mathbf{Q}) \right|^2 , \tag{9.12}$$

where the sum runs over all atoms in one unit cell. This structure factor is non-zero only when Q equals a reciprocal lattice vector.

The textbook expression for the scattering cross section corresponding to one Debye-Scherrer cone reads [Squ78, ch.3.6], with $V = NV_0$ being the total sample volume:

$$\sigma_{\text{cone}} = \frac{V}{V_0^2} \frac{\lambda^3}{4 \sin\theta} \sum_Q |F(Q)|^2. \tag{9.13}$$

For our purpose, this expression should be changed slightly. Firstly, the sum over structure factors for a particular Q is replaced by the sum over essentially different reflections multiplied by their multiplicity, j. Then, a finite packing factor, f, is defined for the powder, and finally, the Debye-Waller factor is multiplied on the elastic cross section to take lattice vibrations into account (no inelastic background is simulated, however). We then reach

$$\sigma_{\text{cone, Q}} = j_Q f \exp(-2W) \frac{V}{V_0^2} \frac{\lambda^3}{4 \sin\theta} |F(Q)|^2 \tag{9.14}$$

$$= f \exp(-2W) \frac{N}{V_0} \frac{4\pi^3}{k^2} \frac{j_Q |F(Q)|^2}{Q} \tag{9.15}$$

in the thin sample approximation. For samples of finite thickness, the beam is being attenuated by the attenuation coefficient

$$\mu_{\text{Q}} = \sigma_{\text{cone,Q}} / V. \tag{9.16}$$

For calibration it may be useful to consider the total intensity scattered into a detector of effective height h, covering only one reflection [Squ78, ch.3.6]. A cut though the Debye-Scherrer cone perpendicular to its axis is a circle. At the distance r from the sample, the radius of this circle is $r \sin(2\theta)$. Thus, the detector (in a small angle approximation) counts a fraction $h/(2\pi r \sin(2\theta))$ of the scattered neutrons, giving a resulting count intensity:

$$I = \Psi \sigma_{\text{cone,Q}} \frac{h}{2\pi r \sin(2\theta)}, \tag{9.17}$$

where Ψ is the flux at the sample position.

For clarity we repeat the meaning and unit of the symbols:

```
\begin{tabular}{ccl}
$\Psi$ & s$^{-1}$m$^{-2}$ & Incoming intensity of neutrons \\
$I$    & s$^{-1}$ & Detected intensity of neutrons \\
$h$    & m        & Height of detector \\
$r$    & m        & Distance from sample to detector \\
$f$    & 1        & Packing factor of the powder \\
$j$    & 1        & Multiplicity of the reflection \\
$V_0$  & m$^{3}$  & Volume of unit cell\\
$|F(\textbf{Q})|^2$ & m$^2$  & Structure factor \\
$\exp(-2W)$ & 1  & Debye-Waller factor \\
$\mu_\textrm{Q}$ & m$^{-1}$ & Linear attenuation factor due to scattering
    from
one powder line. \\
\end{tabular}
```

A powder sample will in general have several allowed reflections $\mathbf{Q}_j$, which will all contribute to the attenuation. These reflections will have different values of $|F(\mathbf{Q}_j)|^2$ (and hence of Q_j), j_j, $\exp(-2W_j)$, and θ_j. The total attenuation through the sample due to scattering is given by $\mu^s = \mu^s_{\text{inc}} + \sum_j \mu^s_j$, where μ^s_{inc} represents the incoherent scattering.

9.3.4. Algorithm

The algorithm of **PowderN** can be summarized as

- Check if the neutron ray intersects the sample (otherwise ignore the following).
- Calculate the attenuation coefficients for scattering and absorption.
- Perform Monte Carlo choices to determine the scattering position, scattering type (coherent/incoherent), and the outgoing direction.
- Perform the necessary weight factor transformation.

9.4. The `Single_crystal` McStas Component

Mosaic single crystal with multiple scattering vectors, optimised for speed with large crystals and many reflections.

Identification

- **Author:** Kristian Nielsen
- **Origin:** Risoe
- **Date:** December 1999

Description

Single crystal with mosaic. Delta-D/D option for finite-size effects. Rectangular geometry. Multiple scattering and secondary extinction included. The mosaic may EITHER be specified isotropic by setting the mosaic input parameter, OR anisotropic by setting the mosaic_a, mosaic_b, and mosaic_c parameters. The crystal lattice can be bent locally, keeping the external geometry unchanged. Curvature is spherical along vertical and horizontal axes.

Speed/stat optimisation using SPLIT In order to dramatically improve the simulation efficiency, we recommend to use a SPLIT keyword on this component (or prior to it), as well as to disable the multiple scattering handling by setting order=1. This is especially powerful for large reflection lists such as with macromolecular proteins. When an incoming particle is identical to the preceeding, reciprocal space initialisation is skipped, and a Monte Carlo choice is done on available reflections from the last repciprocal space calculation! To assist the user in choosing a "relevant" value of the SPLIT, a rolling average of the number of available reflections is calculated and presented in the component output.

Mosacitiy modes: The component features three independent ways of parametrising mosaicity: a) The original algorithm where mosaicity is implemented by extending each reflection by a Gaussian "cigar" in reciprocal space, characterised by the parameters mosaic and delta_d_d. (Also known as "isotropic mosaicity".) b) A similar mode where mosaicities can be non-isotropic and given as the parameters mosaic_a, mosaic_b and mosaic_c, around the unit cell axes. (Also known as "anisotropic mosaicity".) c) Given two "macroscopically"/experimentally measured width/mosaicities of two independent reflections, parametrised by the list mosaic_AB = {mos_a, mos_b, a_h, a_k, a_l, b_h, b_k, b_l}, a set of microscopic mosaicities as in b) are estimated (internally) and applied. (Also known as "phenomenological mosaicity".)

Powder- and PG-mode When these two modes are used (powder=1 or PG=1), a randomised transformation of the particle direction is made before and after scattering, thereby letting the single crystal behave as a crystallite of either a powder (crystallite orientation fully randomised) or pyrolytic graphite (crystallite randomised around the c-axis).

Curved crystal mode The component features a method to curve the lattice planes slightly with respect to the outer geometry of the crystal. The method is implemented as a transformation on the particle direction vector, and should be used only in cases where a) The reflection lattice vector is ~ orthogonal to the crystal surface b) The modelled curvarture is "small" with respect to the crystal surface

Sample shape: Sample shape may be a cylinder, a sphere, a box or any other shape

```
box/plate:       xwidth x yheight x zdepth
cylinder:        radius x yheight
sphere:          radius (yheight=0)
any shape:       geometry=OFF, PLY or STL file
```

The complex geometry option handles any closed non-convex polyhedra. It computes the intersection points of the neutron ray with the object transparently, so that it can be used like a regular sample object. It supports the OFF (incl. NOFF, COFF), PLY and STL file formats. A mesh must be a single, closed object: the neutron path through it is taken from the first entry and exit only, so a second part of the mesh further along the path (disjoint parts, or a concave shape the path re-enters) is not seen. Such files may be generated from XYZ data using: qhull < coordinates.xyz Qx Qv Tv o > geomview.off or powercrust coordinates.xyz and viewed with geomview or java -jar jroff.jar (see below). The default size of the object depends on the OFF file data, but its bounding box may be resized using xwidth,yheight and zdepth.

Crystal definition file format Crystal structure is specified with an ascii data file. Each line contains 4 or more numbers, separated by white spaces:

```
h k l ... F2
```

The first three numbers are the (h,k,l) indices of the reciprocal lattice point, and the 7-th number is the value of the structure factor |F|**2, in barns. The rest of the numbers are not used; the file is in the format output by the Crystallographica program. The reflection list should be ordered by decreasing d-spacing values. Lines begining by '#' are read as comments (ignored). Most sample parameters may be defined from the data file header, following the same mechanism as PowderN.

Current data file header keywords include, for data format specification: #column_h <index of the Bragg Qh column> #column_k <index of the Bragg Qk column> #column_l <index of the Bragg Ql column> #column_F2 <index of the squared str. factor '|F|^2' column [b]> #column_F <index of the structure factor norm '|F|' column> and for material specification: #sigma_abs <value of absorption cross section [barns]> #sigma_inc <value of incoherent cross section [barns]> #Delta_d/d <value of Detla_d/d width for all lines> #lattice_a <value of the a lattice parameter [Å]> #lattice_b <value of the b lattice parameter [Å]> #lattice_c <value of the c lattice parameter [Å]> #lattice_aa <value of the alpha lattice angle [deg]> #lattice_bb <value of the beta lattice angle [deg]> #lattice_cc <value of the gamma lattice angle [deg]>

Last, CIF, FullProf and ShelX files can be read, and converted to F2(hkl) lists if 'cif2hkl' is installed. The CIF2HKL env variable can be used to point to a proper executable, else the McCode, then the system installed versions are used.

NCrystal materials The 'reflections' parameter can also be an NCrystal cfg-string (containing ".ncmat" or "::"), e.g. "stdlib::Al_sg225.ncmat" or "mycrystal.ncmat". The reflections, lattice parameters and cross sections are then taken from NCrystal. The material must have exactly one crystalline phase (but may also have non-crystalline phases), and must not specify an orientation, as that is defined by the component. Only Bragg diffraction, absorption and incoherent scattering are modelled. For the full NCrystal physics, use the NCrystal_sample component, or the NCrystal_process component with Union. See the Component Manual for more defails.

Example: Single_crystal(xwidth=0.01, yheight=0.01, zdepth=0.01, mosaic = 5, reflections="YBaCuO.lau")

A PG graphite crystal plate, cut for (002) reflections Single_crystal(xwidth = 0.002, yheight = 0.1, zdepth = 0.1, mosaic = 30, reflections = "C_graphite.lau",

```
ax=0,      ay=2.14,   az=-1.24,
bx = 0,    by = 0,    bz =  2.47,
cx = 6.71, cy = 0,    cz =  0)
```

A leucine protein, without multiple scattering Single_crystal(xwidth=0.005, yheight=0.005, zdepth=0.005, mosaic = 5, reflections="leucine.lau", order=1)

A Vanadium incoherent elastic scattering with multiple scattering Single_crystal(xwidth=0.01, yheight=0.01, zdepth=0.01, reflections="", sigma_abs=5.08, sigma_inc=4.935,

```
ax=3.0282, by=3.0282, cz=3.0282/2)
```

Also, always use a non-zero value of delta_d_d.

%VALIDATION: This component has been validated.

This sample component can advantageously benefit from the SPLIT feature, e.g. SPLIT COMPONENT sx = Single_crystal(...)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
reflections	string	File name containing structure factors of reflections (LAZ LAU CIF, FullProf, ShelX, NCrystal). Use empty ("") or NULL for incoherent scattering only	0

Name	Unit	Description	Default
geometry	str	Name of an OFF, PLY or STL file for complex geometry. An OFF file may be generated from XYZ coordinates using qhull/powercrust	0
mosaic_AB	arc_minutes, arc_minutes, 1, 1, 1, 1, 1, 1	In Plane mosaic rotation and plane vectors (anisotropic), mosaic_A, mosaic_B, A_h,A_k,A_l, B_h,B_k,B_l. Puts the crystal in the in-plane mosaic state. Vectors A and B define plane in which the crystal roation is defined, and mosaic_A, mosaic_B, denotes the resp. mosaicities (gaussian RMS) with respect to the two reflections chosen by A and B (Miller indices).	{0,0, 0,0,0, 0,0,0}
xwidth	m	Width of crystal	0
yheight	m	Height of crystal	0
zdepth	m	Depth of crystal (no extinction simulated)	0
radius	m	Outer radius of sample in (x,z) plane	0
delta_d_d	1	Lattice spacing variance, gaussian RMS	1e-4
mosaic	arc minutes	Crystal mosaic (isotropic), gaussian RMS. Puts the crystal in the isotropic mosaic model state, thus disregarding other mosaicity parameters.	-1
mosaic_a	arc minutes	Horizontal (rotation around lattice vector a) mosaic (anisotropic), gaussian RMS. Put the crystal in the anisotropic crystal vector state. I.e. model mosaicity through rotation around the crystal lattice vectors. Has precedence over in-plane mosaic model.	-1
mosaic_b	arc minutes	Vertical (rotation around lattice vector b) mosaic (anisotropic), gaussian RMS.	-1
mosaic_c	arc minutes	Out-of-plane (Rotation around lattice vector c) mosaic (anisotropic), gaussian RMS	-1
recip_cell	1	Choice of direct/reciprocal (0/1) unit cell definition	0

Name	Unit	Description	Default
barns	1	Flag to indicate if |F|^2 from 'reflections' is in barns or fm^2. barns=1 for laz and isotropic constant elastic scattering (reflections=NULL), barns=0 for lau type files	0
ax	Å or $Å^{-1}$	Coordinates of first (direct/recip) unit cell vector	0
ay	Å or $Å^{-1}$	a on y axis	0
az	Å or $Å^{-1}$	a on z axis	0
bx	Å or $Å^{-1}$	Coordinates of second (direct/recip) unit cell vector	0
by	Å or $Å^{-1}$	b on y axis	0
bz	Å or $Å^{-1}$	b on z axis	0
cx	Å or $Å^{-1}$	Coordinates of third (direct/recip) unit cell vector	0
cy	Å or $Å^{-1}$	c on y axis	0
cz	Å or $Å^{-1}$	c on z axis	0
p_transmit	1	Monte Carlo probability for neutrons to be transmitted without any scattering. Used to improve statistics from weak reflections	0.001
sigma_abs	barns	Absorption cross-section per unit cell at 2200 m/s	0
sigma_inc	barns	Incoherent scattering cross-section per unit cell Use -1 to inactivate	0
aa	deg	Unit cell angles alpha, beta and gamma. Then uses norms of vectors a,b and c as lattice parameters	0
bb	deg	Beta angle	0
cc	deg	Gamma angle	0
order	1	Limit multiple scattering up to given order (0: all, 1: first, 2: second, ...)	0
extra_order	1	When using order, allow additional multiple scattering without coherent scattering, sensible with very large unit cells (0: disable, 1: one extra, 2: two extra, ...)	0
RX	m	Radius of horizontal along X lattice curvature. flat for 0	0
RY	m	Radius of vertical along Y lattice curvature. flat for 0	0

Name	Unit	Description	Default
powder	1	Flag to indicate powder mode, for simulation of Debye-Scherrer cones via random crystallite orientation. A powder texture can be approximated with 0<powder<1	0
PG	1	Flag to indicate "Pyrolytic Graphite" mode, only meaningful with choice of Graphite.lau, models PG crystal. A powder texture can be approximated with 0<PG<1 with main axis on 'c'	0

Links

- Component source code found in file `Single_crystal.comp`.
- See ICSD Inorganic Crystal Structure Database
- Cross sections for single elements
- Cross sections for compounds
- Web Elements
- Fullprof powder refinement
- Crystallographica software
- Geomview and Object File Format (OFF)
- Java version of Geomview (display only) jroff.jar
- qhull
- powercrust

The single crystal component

The **Single_crystal** component models a thick, flat single crystal with multiple scattering and absorption with elastic coherent scattering. An elastic incoherent background may also be simulated. It may be used to describe samples for diffraction, but also for accurate monochromator descriptions. The component is currently under further review. The current documentation is outdated, especially with respect to the model of crystal mosaicity.

The input parameters for the component are *xwidth*, *yheight*, and *zdepth* to define the dimensions of the crystal in meters (area is centered); *delta_ d_ d* to give the value of $\Delta d/d$ (no unit); (ax, ay, az), (bx, by, bz), and (cx, cy, cz) to define the axes of the direct lattice

of the crystal (the sides of the unit cell) in units of Ångström; and *reflections*, a string giving the name of the file with the list of structure factors to consider. The mosaic is specified *either* isotropically as *mosaic*, *or* anisotropically as *mosaic_ a* (rotation around lattice vector $\vec{a}$), *mosaic_ b* (rotation around lattice vector $\vec{b}$), and *mosaic_ c* (rotation around lattice vector $\vec{c}$); in all cases in units of full-width-half-maximum minutes of arc.

Optionally, the absorption cross-section at 2200 m/s and the incoherent cross-section may be given as *sigma_ abs* and *sigma_ inc* (in barns), with default of zero; and *p_ transmit* may be assigned a fixed Monte Carlo probability for transmission through the crystal without any interaction.

The user must specify a list of reciprocal lattice vectors $\boldsymbol{\tau}$ to consider along with their structure factors $|F_{\boldsymbol{\tau}}|^2$. The user must also specify the coordinates (in direct space) of the unit cell axes $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$, from which the reciprocal lattice will be computed. See section 9.4.5 for file format specifications.

In addition to coherent scattering, **Single_crystal** also handles incoherent scattering and absorption. The incoherent scattering cross-section is supplied by the user as a constant σ_{inc}. The absorption cross-section is supplied by the user at 2200 m/s, so the actual cross-section for a neutron of velocity v is $\sigma_{\text{abs}} = \sigma_{2200}\frac{2200 \text{ m/s}}{v}$.

9.4.1. The physical model

The textbook expression for the scattering cross-section of a crystal is [Squ78, ch.3]:

$$\left(\frac{d\sigma}{d\Omega}\right)_{\text{coh.el.}} = N\frac{(2\pi)^3}{V_0}\sum_{\boldsymbol{\tau}}\delta(\boldsymbol{\tau}-\boldsymbol{\kappa})|F_{\boldsymbol{\tau}}|^2 \tag{9.18}$$

Here $|F_{\boldsymbol{\tau}}|^2$ is the structure factor (defined in section 9.3), N is the number of unit cells, V_0 is the volume of an individual unit cell, and $\boldsymbol{\kappa}(= \mathbf{k}_i - \mathbf{k}_f)$ is the scattering vector. $\delta(\boldsymbol{x})$ is a 3-dimensional delta function in reciprocal space, so for given incoming wave vector $\mathbf{k}_i$ and lattice vector $\boldsymbol{\tau}$, only a single final wave vector $\mathbf{k}_f$ is allowed. In general, this wavevector will not fulfill the conditions for elastic scattering ($k_f = k_i$). In a real crystal, however, reflections are not perfectly sharp. Because of imperfection and finite-size effects, there will be a small region around $\boldsymbol{\tau}$ in reciprocal space of possible scattering vectors.

Single_crystal simulates a crystal with a mosaic spread η and a lattice plane spacing uncertainty $\Delta d/d$. In such crystals the reflections will not be completely sharp; there will be a small region around each reciprocal lattice point of the crystal that contains valid scattering vectors.

We model the mosaicity and $\Delta d/d$ of the crystal with 3-dimensional Gaussian functions in reciprocal space (see figure 9.4). Two of the axes of the Gaussian are perpendicular to the reciprocal lattice vector $\boldsymbol{\tau}$ and model the mosaicity. The third one is parallel to $\boldsymbol{\tau}$ and models $\Delta d/d$. We assume that the mosaicity is small so that the possible directions of the scattering vector may be approximated with a Gaussian in rectangular coordinates.

If the mosaic is isotropic (the same in all directions), the two Gaussian axes perpendicular to $\boldsymbol{\tau}$ are simply arbitrary normal vectors of equal length given by the mosaic.

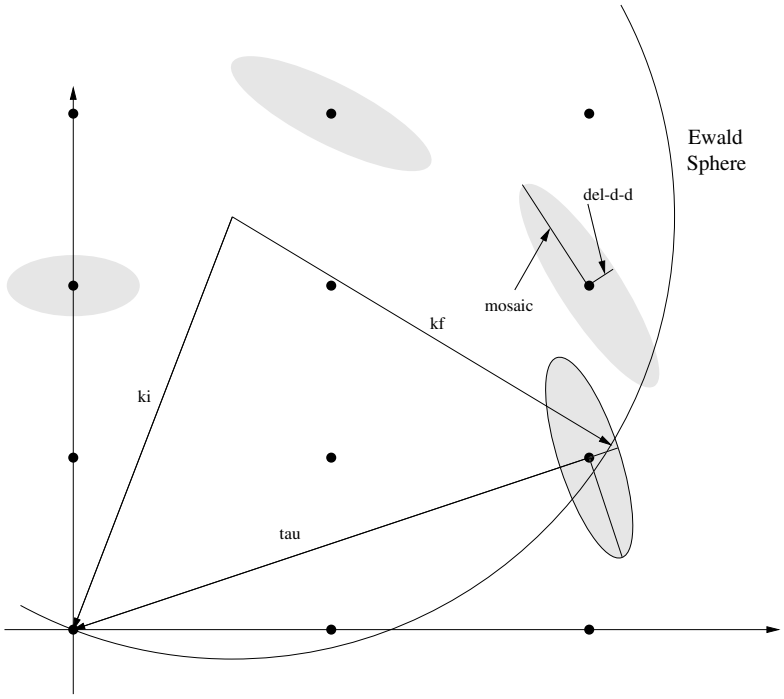


Figure 9.4.: Ewald sphere construction for a single neutron showing the Gaussian broadening of reciprocal lattice points in their local coordinate system.

But if the mosaic is anisotropic, the two perpendicular axes will in general be different for each scattering vector. In the absence of anything better, **Single_crystal** uses a model which is at least mathematically plausible and which works as expected in the two common cases: (1) isotropic mosaic, and (2) two mosaic directions ("horizontal and vertical mosaic") perpendicular to a scattering vector.

The basis for the model is a three-dimensional Gaussian distribution in Euler angles giving the orientation probability distribution for the micro-crystals; that is, the misorientation is given by small rotations around the X, Y, and Z axes, with the rotation angles having (in general different) Gaussian probability distributions. For given scattering vector $\boldsymbol{\tau}$, a rotation of the micro-crystals around an axis parallel to $\boldsymbol{\tau}$ has no effect on the direction of the scattering vector. Suppose we form the intersection between the three-dimensional Gaussian in Euler angles and a plane through the origin perpendicular to $\boldsymbol{\tau}$. This gives a two-dimensional Gaussian, say with axes defined by unit vectors $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ and mosaic widths η_1 and η_2.

We now let the mosaic for $\boldsymbol{\tau}$ be defined by rotations around $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ with angles having Gaussian distributions of widths η_1 and η_2. Since $\boldsymbol{g}_1$, $\boldsymbol{g}_2$, and $\boldsymbol{\tau}$ are perpendicular, a small rotation of $\boldsymbol{\tau}$ around $\boldsymbol{g}_1$ will change $\boldsymbol{\tau}$ in the direction of $\boldsymbol{g}_2$. The two axes of the Gaussian mosaic in reciprocal space that are perpendicular to $\boldsymbol{\tau}$ will thus be given by $\tau\eta_2\boldsymbol{g}_1$ and $\tau\eta_1\boldsymbol{g}_2$.

We now derive a quantitative expression for the scattering cross-section of the crystal in the model. For this, we introduce a *local coordinate system* for each reciprocal lattice point $\boldsymbol{\tau}$ and use $\boldsymbol{x}$ for vectors written in local coordinates. The origin is $\boldsymbol{\tau}$, the first axis is parallel to $\boldsymbol{\tau}$ and the other two axes are perpendicular to $\boldsymbol{\tau}$. In the local coordinate system, the 3-dimensional Gaussian is given by

$$G(x_1, x_2, x_3) = \frac{1}{(\sqrt{2\pi})^3} \frac{1}{\sigma_1 \sigma_2 \sigma_3} e^{-\frac{1}{2}(\frac{x_1^2}{\sigma_1^2} + \frac{x_2^2}{\sigma_2^2} + \frac{x_3^2}{\sigma_3^2})} \tag{9.19}$$

The axes of the Gaussian are $\sigma_1 = \tau \Delta d/d$ and $\sigma_2 = \sigma_3 = \eta\tau$. Here we used the assumption that η is small, so that $\tan\eta \approx \eta$ (with η given in radians). By introducing the diagonal matrix

$$D = \begin{pmatrix} \frac{1}{2}\sigma_1^2 & 0 & 0 \\ 0 & \frac{1}{2}\sigma_2^2 & 0 \\ 0 & 0 & \frac{1}{2}\sigma_3^2 \end{pmatrix}$$

equation (9.19) can be written as

$$G(\boldsymbol{x}) = \frac{1}{(\sqrt{2\pi})^3} \frac{1}{\sigma_1 \sigma_2 \sigma_3} e^{-\boldsymbol{x}^{\mathrm{T}} D \boldsymbol{x}} \tag{9.20}$$

again with $\boldsymbol{x} = (x_1, x_2, x_3)$ written in local coordinates.

To get an expression in the coordinates of the reciprocal lattice of the crystal, we introduce a matrix U such that if $\boldsymbol{y} = (y_1, y_2, y_3)$ are the global coordinates of a point in the crystal reciprocal lattice, then $U(\boldsymbol{y} + \boldsymbol{\tau})$ are the coordinates in the local coordinate system for $\boldsymbol{\tau}$. The matrix U is given by

$$U^{\mathrm{T}} = (\hat{u}_1, \hat{u}_2, \hat{u}_3),$$

where $\hat{u}_1$, $\hat{u}_2$, and $\hat{u}_3$ are the axes of the local coordinate system, written in the global coordinates of the reciprocal lattice. Thus $\hat{u}_1 = \boldsymbol{\tau}/\tau$, and $\hat{u}_2$ and $\hat{u}_3$ are unit vectors perpendicular to $\hat{u}_1$ and to each other. The matrix U is unitarian, that is $U^{-1} = U^{\mathrm{T}}$. The translation between global and local coordinates is

$$\boldsymbol{x} = U(\boldsymbol{y} + \boldsymbol{\tau}) \qquad \boldsymbol{y} = U^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{\tau}$$

The expression for the 3-dimensional Gaussian in global coordinates is

$$G(\boldsymbol{y}) = \frac{1}{(\sqrt{2\pi})^3} \frac{1}{\sigma_1 \sigma_2 \sigma_3} e^{-(U(\boldsymbol{y}+\boldsymbol{\tau}))^{\mathrm{T}} D (U(\boldsymbol{y}+\boldsymbol{\tau}))} \tag{9.21}$$

The elastic coherent cross-section is then given by

$$\left(\frac{d\sigma}{d\Omega}\right)_{\text{coh.el.}} = N \frac{(2\pi)^3}{V_0} \sum_{\boldsymbol{\tau}} G(\boldsymbol{\tau} - \boldsymbol{\kappa}) |F_{\boldsymbol{\tau}}|^2 \tag{9.22}$$

9.4.2. The algorithm

The overview of the algorithm used in the Single_crystal component is as follows:

1. Check if the neutron intersects the crystal. If not, no action is taken.
2. Search through a list of reciprocal lattice points of interest, selecting those that are close enough to the Ewald sphere to have a non-vanishing scattering probability. From these, compute the total coherent cross-section σ_{coh} (see below), the absorption cross-section $\sigma_{\mathrm{abs}} = \sigma_{2200}\frac{2200\ \mathrm{m/s}}{v}$, and the total cross-section $\sigma_{\mathrm{tot}} = \sigma_{\mathrm{coh}} + \sigma_{\mathrm{inc}} + \sigma_{\mathrm{abs}}$.
3. The transmission probability is $\exp(-\frac{\sigma_{\mathrm{tot}}}{V_0}\ell)$ where ℓ is the length of the flight path through the crystal. A Monte Carlo choice is performed to determine whether the neutron is transmitted. Optionally, the user may set a fixed Monte Carlo probability for the first scattering event, for example to boost the statistics for a weak reflection.
4. For non-transmission, the position at which the neutron will interact is selected from an exponential distribution. A Monte Carlo choice is made of whether to scatter coherently or incoherently. Absorption is treated by weight adjustment (see below).
5. For incoherent scattering, the outgoing wave vector $\boldsymbol{k}_{\mathrm{f}}$ is selected with a random direction.
6. For coherent scattering, a reciprocal lattice vector is selected by a Monte Carlo choice, and $\boldsymbol{k}_{\mathrm{f}}$ is found (see below).
7. Adjust the neutron weight as dictated by the Monte Carlo choices made.
8. Repeat from (2) until the neutron is transmitted (to simulate multiple scattering).

For point 2, the distance *dist* between a reciprocal lattice point and the Ewald sphere is considered small enough to allow scattering if it is less than five times the maximum axis of the Gaussian, $dist \leq 5\max(\sigma_1, \sigma_2, \sigma_3)$.

9.4.3. Choosing the outgoing wave vector

The final wave vector $\boldsymbol{k}_{\mathrm{f}}$ must lie on the intersection between the Ewald sphere and the Gaussian ellipsoid. Since η and $\Delta d/d$ are assumed small, the intersection can be approximated with a plane tangential to the sphere, see figure 9.5. The tangential point is taken to lie on the line between the center of the Ewald sphere $-\boldsymbol{k}_{\mathrm{i}}$ and the reciprocal lattice point $\boldsymbol{\tau}$. Since the radius of the Ewald sphere is k_{i}, this point is

$$\boldsymbol{o} = (k_{\mathrm{i}}/\rho - 1)\boldsymbol{\rho} - \boldsymbol{\tau}$$

where $\boldsymbol{\rho} = \boldsymbol{k}_{\mathrm{i}} - \boldsymbol{\tau}$.

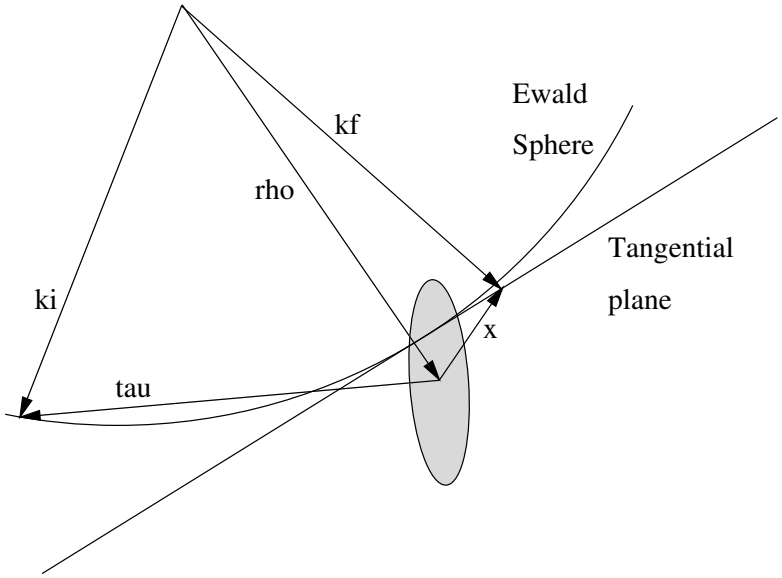


Figure 9.5.: The scattering triangle in the single crystal.

The equation for the plane is

$$\boldsymbol{P}(\boldsymbol{t}) = \boldsymbol{o} + B\boldsymbol{t}, \qquad \boldsymbol{t} \in \mathbb{R}^2 \tag{9.23}$$

Here $B = (\boldsymbol{b}_1, \boldsymbol{b}_2)$ is a 3×2 matrix with the two generators for the plane $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$. These are (arbitrary) unit vectors in the plane, being perpendicular to each other and to the plane normal $\boldsymbol{n} = \boldsymbol{\rho}/\rho$.

Each $\boldsymbol{t}$ defines a potential final wave vector $\boldsymbol{k}_\mathrm{f}(\boldsymbol{t}) = \boldsymbol{k}_\mathrm{i} + \boldsymbol{P}(\boldsymbol{t})$. The value of the 3-dimensional Gaussian for this $\boldsymbol{k}_\mathrm{f}$ is

$$G(\boldsymbol{x}(\boldsymbol{t})) = \frac{1}{(\sqrt{2\pi})^3} \frac{1}{\sigma_1\sigma_2\sigma_3} e^{-\boldsymbol{x}(\boldsymbol{t})^\mathrm{T} D \boldsymbol{x}(\boldsymbol{t})} \tag{9.24}$$

where $\boldsymbol{x}(\boldsymbol{t}) = \boldsymbol{\tau} - (\boldsymbol{k}_\mathrm{i} - \boldsymbol{k}_\mathrm{f}(\boldsymbol{t}))$ is given in local coordinates for $\boldsymbol{\tau}$. It can be shown that equation (9.24) can be re-written as

$$G(\boldsymbol{x}(\boldsymbol{t})) = \frac{1}{(\sqrt{2\pi})^3} \frac{1}{\sigma_1\sigma_2\sigma_3} e^{-\alpha} e^{-(\boldsymbol{t}-\boldsymbol{t}_0)^\mathrm{T} M (\boldsymbol{t}-\boldsymbol{t}_0)} \tag{9.25}$$

where $M = B^\mathrm{T} D B$ is a 2×2 symmetric and positive definite matrix, $\boldsymbol{t}_0 = -M^{-1} B^\mathrm{T} D \boldsymbol{o}$ is a 2-vector, and $\alpha = -\boldsymbol{t}_0^\mathrm{T} M \boldsymbol{t}_0 + \boldsymbol{o}^\mathrm{T} D \boldsymbol{o}$ is a real number. Note that this is a two-dimensional Gaussian (not necessarily normalized) in $\boldsymbol{t}$ with center $\boldsymbol{t}_0$ and axis defined by M.

To choose $\boldsymbol{k}_\mathrm{f}$ we sample $\boldsymbol{t}$ from the 2-dimensional Gaussian distribution (9.25). To do this, we first construct the Cholesky decomposition of the matrix $(\frac{1}{2}M^{-1})$. This gives a

2×2 matrix L such that $LL^{\mathrm{T}} = \frac{1}{2}M^{-1}$ and is possible since M is symmetric and positive definite. It is given by

$$L = \begin{pmatrix} \sqrt{\nu_{11}} & 0 \\ \frac{\nu_{12}}{\sqrt{\nu_{11}}} & \sqrt{\nu_{22} - \frac{\nu_{12}^2}{\nu_{11}}} \end{pmatrix} \qquad \text{where } \frac{1}{2}M^{-1} = \begin{pmatrix} \nu_{11} & \nu_{12} \\ \nu_{12} & \nu_{22} \end{pmatrix}$$

Now let $\boldsymbol{g} = (g_1, g_2)$ be two random numbers drawn form a Gaussian distribution with mean 0 and standard deviation 1, and let $\boldsymbol{t} = L\boldsymbol{g} + \boldsymbol{t}_0$. The probability of a particular $\boldsymbol{t}$ is then

$$\begin{aligned} P(\boldsymbol{t})d\boldsymbol{t} &= \frac{1}{2\pi} e^{-\frac{1}{2}\boldsymbol{g}^{\mathrm{T}}\boldsymbol{g}} d\boldsymbol{g} \\ &= \frac{1}{2\pi}\frac{1}{\det L} e^{-\frac{1}{2}(L^{-1}(\boldsymbol{t}-\boldsymbol{t}_0))^{\mathrm{T}}(L^{-1}(\boldsymbol{t}-\boldsymbol{t}_0))} d\boldsymbol{t} \\ &= \frac{1}{2\pi}\frac{1}{\det L} e^{-(\boldsymbol{t}-\boldsymbol{t}_0)^{\mathrm{T}}M(\boldsymbol{t}-\boldsymbol{t}_0)} d\boldsymbol{t} \end{aligned} \tag{9.26}$$

where we used that $\boldsymbol{g} = L^{-1}(\boldsymbol{t} - \boldsymbol{t}_0)$ so that $d\boldsymbol{g} = \frac{1}{\det L} d\boldsymbol{t}$. This is just the normalized form of (9.25). Finally we set $\boldsymbol{k}'_{\mathrm{f}} = \boldsymbol{k}_{\mathrm{i}} + \boldsymbol{P}(\boldsymbol{t})$ and $\boldsymbol{k}_{\mathrm{f}} = (k_{\mathrm{i}}/k'_f)\boldsymbol{k}'_{\mathrm{f}}$ to normalize the length of $\boldsymbol{k}_{\mathrm{f}}$ to correct for the (small) error introduced by approximating the Ewald sphere with a plane.

9.4.4. Computing the total coherent cross-section

To determine the total coherent scattering cross-section, the differential cross-section must be integrated over the Ewald sphere:

$$\sigma_{\mathrm{coh}} = \int_{\mathrm{Ewald}} \left(\frac{d\sigma}{d\Omega}\right)_{\mathrm{coh.el.}} d\Omega$$

For small mosaic we may approximate the sphere with the tangential plane, and we thus get from (9.22) and (9.25):

$$\begin{aligned} \sigma_{\mathrm{coh},\boldsymbol{\tau}} &= \int N\frac{(2\pi)^3}{V_0} G(\boldsymbol{\tau}-\boldsymbol{\kappa})|F_{\boldsymbol{\tau}}|^2 d\Omega \\ &= \frac{1}{\boldsymbol{k}_i^2} N\frac{(2\pi)^3}{V_0}\frac{1}{(\sqrt{2\pi})^3}\frac{e^{-\alpha}}{\sigma_1\sigma_2\sigma_3}|F_{\boldsymbol{\tau}}|^2 \int e^{-(\boldsymbol{t}-\boldsymbol{t}_0)^{\mathrm{T}}M(\boldsymbol{t}-\boldsymbol{t}_0)} d\boldsymbol{t} \\ &= \det(L)\frac{1}{\boldsymbol{k}_i^2} N\frac{(2\pi)^{3/2}}{V_0}\frac{e^{-\alpha}}{\sigma_1\sigma_2\sigma_3}|F_{\boldsymbol{\tau}}|^2 \int e^{-\frac{1}{2}\boldsymbol{g}^{\mathrm{T}}\boldsymbol{g}} d\boldsymbol{g} \\ &= 2\pi\det(L)\frac{1}{\boldsymbol{k}_i^2} N\frac{(2\pi)^{3/2}}{V_0}\frac{e^{-\alpha}}{\sigma_1\sigma_2\sigma_3}|F_{\boldsymbol{\tau}}|^2 \\ &= \frac{\det(L)}{\boldsymbol{k}_i^2} N\frac{(2\pi)^{5/2}}{V_0}\frac{e^{-\alpha}}{\sigma_1\sigma_2\sigma_3}|F_{\boldsymbol{\tau}}|^2 \\ \sigma_{\mathrm{coh}} &= \sum_{\boldsymbol{\tau}} \sigma_{\mathrm{coh},\boldsymbol{\tau}} \end{aligned} \tag{9.27}$$

As before, we let $\boldsymbol{g} = L^{-1}(\boldsymbol{t} - \boldsymbol{t}_0)$ so that $d\boldsymbol{t} = \det(L)d\boldsymbol{g}$.

Neutron weight factor adjustment We now calculate the correct neutron weight adjustment for the Monte Carlo choices made. In three cases is a Monte Carlo choice made with a probability different from the probability of the corresponding physical event: When deciding whether to transmit the neutron or not, when simulating absorption, and when selecting the reciprocal lattice vector $\boldsymbol{\tau}$ to scatter from.

If the user has choosen a fixed transmission probability $f(\text{ transmit}) = p_{\text{transmit}}$, the neutron weight must be adjusted by

$$\pi(\text{transmit}) = \frac{P(\text{transmit})}{f(\text{transmit})}$$

where $P(\text{transmit}) = \exp(-\frac{\sigma_{\text{tot}}}{V_0}\ell)$ is the physical transmission probability. Likewise, for non-transmission the adjustment is

$$\pi(\text{no transmission}) = \frac{1 - P(\text{transmit})}{1 - f(\text{transmit})}.$$

Absorption is never explicitly simulated, so the Monte Carlo probability of coherent or incoherent scattering is $f(\text{coh}) + f(\text{inc}) = 1$. The physical probability of coherent or incoherent scattering is

$$P(\text{coh}) + P(\text{inc}) = \frac{\sigma_{\text{coh}} + \sigma_{\text{inc}}}{\sigma_{\text{tot}}},$$

so again a weight adjustment $\pi(\text{coh|inc}) = \Pi(\text{ coh|inc})/f(\text{coh|inc})$ is needed.

When choosing the reciprocal lattice vector $\boldsymbol{\tau}$ to scatter from, the relative probability for $\boldsymbol{\tau}$ is $r_{\boldsymbol{\tau}} = \sigma_{\text{coh},\boldsymbol{\tau}}/|F_{\boldsymbol{\tau}}|^2$. This is done to get better statistics for weak reflections. The Monte Carlo probability for the reciprocal lattice vector $\boldsymbol{\tau}$ is thus

$$f(\boldsymbol{\tau}) = \frac{r_{\boldsymbol{\tau}}}{\sum_{\boldsymbol{\tau}} r_{\boldsymbol{\tau}}}$$

whereas the physical probability is $P(\boldsymbol{\tau}) = \sigma_{\text{coh},\boldsymbol{\tau}}/\sigma_{\text{coh}}$. A weight adjustment is thus needed of

$$\pi(\boldsymbol{\tau}) = \frac{P(\boldsymbol{\tau})}{f(\boldsymbol{\tau})} = \frac{\sigma_{\text{coh},\boldsymbol{\tau}} \sum_{\boldsymbol{\tau}} r_{\boldsymbol{\tau}}}{\sigma_{\text{coh}}\, r_{\boldsymbol{\tau}}}.$$

In most cases, however, only one reflection is possible, whence $\pi = 1$.

9.4.5. Implementation details

The equations describing **Single_crystal** are quite complex, and consequently the code is fairly sizeable. Most of it is just the expansion of the vector and matrix equations in individual coordinates, and should thus be straightforward to follow.

The implementation pre-computes a lot of the necessary values in the `INITIALIZE` section. It is thus actually very efficient despite the complexity. If the list of reciprocal lattice points is big, however, the search through the list will be slow. The precomputed data is stored in the structures `hkl_info` and in an array of `hkl_data` structures (one for each reciprocal lattice point in the list). In addition, for every neutron event an array

of `tau_data` is computed with one element for each reciprocal lattice point close to the Ewald sphere. Except for the search for possible $\boldsymbol{\tau}$ vectors, all computations are done in local coordinates using the matrix U to do the necessary transformations.

The list of reciprocal lattice points is specified in an ASCII data file. Each line contains seven numbers, separated by white space. The first three numbers are the (h, k, l) indices of the reciprocal lattice point, and the last number is the value of the structure factor $|F_{\boldsymbol{\tau}}|^2$, in barns. The middle three numbers are not used and may be omitted; they are nevertheless recommended since this makes the file format compatible with the output from the Crystallographica program [Cry]. Any line beginning with any character of `#;/%` is considered to be a comment, and lines which can not be read as vectors/matrices are ignored.

The column signification may also explicitely be set in the data file header using any of the lines:

```
#column_h <index of the Bragg Qh column>
#column_k <index of the Bragg Qk column>
#column_l <index of the Bragg Ql column>
#column_F2 <index of the squared str. factor '|F|^2' column [b]>
#column_F  <index of the structure factor norm '|F|' column>
```

Other component parameters may as well be specified in the data file header with lines e.g.:

```
#sigma_abs <value of Absorption cross section [barns]>
#sigma_inc <value of Incoherent cross section [barns]>
#Delta_d/d <value of Delta_d/d width for all lines>
#lattice_a <value of the a lattice parameter [Angs]>
#lattice_a <value of the b lattice parameter [Angs]>
#lattice_a <value of the c lattice parameter [Angs]>
#lattice_aa <value of the alpha lattice angle [deg]>
#lattice_bb <value of the beta  lattice angle [deg]>
#lattice_cc <value of the gamma lattice angle [deg]>
```

Example data `*.lau` files are given in directory `MCSTAS/data`.

These files contain an extensive self-documented header defining most the sample parameters, so that only the file name and mosaicity should be given to the component:

```
Single_crystal(xwidth=0.01, yheight=0.01, zdepth=0.01,
  mosaic = 5, reflections="YBaCuO.lau")
```

Powder files from ICSD/LAZY [Ics] and Fullprof [Ful] may also be used (see Table 1.2, page 24). We do not recommend to use these as the equivalent $\vec{q}$ vectors are superposed, not all Bragg spots will be simulated, and the intensity will not be scaled by the multiplicity for each spot.

9.5. The `Sans_spheres` McStas Component

Sample for Small Angle Neutron Scattering - hard spheres in thin solution, mono disperse.

Identification

- **Author:** P. Willendrup, K. Lefmann, L. Arleth
- **Origin:** Risoe
- **Date:** 19.12.2003

Description

Sample for use in a SANS instrument, models hard, mono disperse spheres in thin solution. The shape of the sample may be a filled box with dimensions xwidth, yheight, zdepth, a cylinder with dimensions radius and yheight, a filled sphere with radius.

Example: Sans_spheres(R = 100, Phi = 1e-3, Delta_rho = 0.6, sigma_abs = 50, xwidth=0.01, yheight=0.01, zdepth=0.005)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
R	Å	Radius of scattering hard spheres	100
Phi	1	Particle volume fraction	1e-3
Delta_rho	fm/Å^3	Excess scattering length density	0.6
sigma_abs	m^{-1}	Absorption cross section density at 2200 m/s	0.05
xwidth	m	horiz. dimension of sample, as a width	0
yheight	m	vert . dimension of sample, as a height for cylinder/box	0
zdepth	m	depth of sample	0
radius	m	Outer radius of sample in (x,z) plane for cylinder/sphere	0
target_x	m	position of target to focus at	0
target_y	m	position of target to focus at	0
target_z	m	position of target to focus at	6
target_index	1	Relative index of component to focus at, e.g. next is +1	0
focus_xw	m	horiz. dimension of a rectangular area	0
focus_yh	m	vert. dimension of a rectangular area	0
focus_aw	deg	horiz. angular dimension of a rectangular area	0

Name	Unit	Description	Default
focus_ah	deg	vert. angular dimension of a rectangular area	0
focus_r	m	Detector (disk-shaped) radius	0

Links

- Component source code found in file `Sans_spheres.comp`.
- The test/example instrument SANS.instr.

A sample of hard spheres for small-angle scattering

The component **Sans_spheres** models a sample of small independent spheres of radius R, which are uniformly distributed in a rectangular volume $x_w \times y_h \times z_t$ with a volume fraction ϕ. The absorption cross section density for the spheres is σ_a (in units of m^{-1}), specified for neutrons at 2200 m/s. Absorption and incoherent scattering from the medium is neglected. The difference in scattering length density (the contrast) between the hard spheres and the medium is called $\Delta\rho$. d denotes the distance to the (presumed circular) SANS detector of radius R.

A usage example of this component can be found in the `Neutron site/tests/SANS` instrument from the `mcgui`.

9.5.1. Small-angle scattering cross section

The neutron intensity scattered into a solid angle $\Delta\Omega$ for a flat isotropic SANS sample in transmission geometry is given by [DL03]:

$$I_s(q) = \Psi \Delta\Omega T A z_{\max} \frac{d\sigma_v}{d\Omega}(q), \tag{9.28}$$

where Ψ is the neutron flux, T is the sample transmission, A is the illuminated sample area, and $z_{\max}$ the length of the neutron path through the sample.

In this component, we consider only scattering from a thin solution of monodisperse hard spheres of radius R, where the volume-specific scattering cross section is given by [DL03]

$$\frac{d\sigma_v}{d\Omega}(q) = n(\Delta\rho)^2 V^2 f(q), \tag{9.29}$$

where $f(q) = \left(3\frac{\sin(qR)-qR\cos(qR)}{(qR)^3}\right)^2$, n is the number density of spheres, and $V = 4/3\pi R^3$ is the sphere volume. (The density is thus $n = \phi/V$.)

Multiple scattering is ignored.

9.5.2. Algorithm

All neutrons, which hit the sample volume, are scattered. (Hence, no direct beam is simulated.) For scattered neutrons, the following steps are taken:

1. Choose a value of q uniformly in the interval $[0; q_{\max}]$.
2. Choose a polar angle, α, for the **q**-vector uniformly in $[0; \pi]$.
3. Scatter the neutron according to (q, α).
4. Calculate and apply the correct weight factor correction.

9.5.3. Calculating the weight factor

The scattering position is found by a Monte Carlo choice uniformly along the whole (unscattered) beam path with the sample, length l_{full}, giving $f_l = 1/l_{\text{full}}$. The direction focusing on the detector gives (in an small angle approximation) $f_\Omega = d^2/(\pi R_{\text{det}}^2)$.

Hence, the total weight tranformation factor becomes

$$\pi_j = l_{\text{full}}(\pi R_{\text{det}}^2/d^2)/(4\pi)n(\Delta\rho)^2V^2f(q)\exp(-\mu_a l), \tag{9.30}$$

where μ_a is the linear attenuation factor due to absorption and l is the total neutron path length within the sample.

This component does NOT simulate absolute intensities. This latter depends on the detector parameters.

Some alternative implementations exist as contributed components.

The `SANS` test/example instrument exists in the distribution for this component.

9.6. The `Phonon_simple` McStas Component

A sample for phonon scattering based on cross section expressions from Squires, Ch.3. Possibility for adding an (unphysical) bandgap.

Identification

- **Author:** Kim Lefmann
- **Origin:** Risoe
- **Date:** 04.02.04

Description

Single-cylinder shape. Absorption included. No multiple scattering. No incoherent scattering emitted. No attenuation from coherent scattering. No Bragg scattering. fcc crystal n.n. interactions only One phonon branch only -> phonon polarization not accounted for. Bravais lattice only. (i.e. just one atom per unit cell)

Algorithm: 0. Always perform the scattering if possible (otherwise ABSORB) 1. Choose direction within a focusing solid angle 2. Calculate the zeros of (E_i-E_f-hbar omega(kappa)) as a function of k_f 3. Choose one value of k_f (always at least one is possible!) 4. Perform the correct weight transformation

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	m	Outer radius of sample in (x,z) plane	
yheight	m	Height of sample in y direction	
sigma_abs	barns	Absorption cross section at 2200 m/s per atom	
sigma_inc	barns	Incoherent scattering cross section per atom	
a	Å	fcc Lattice constant	
b	fm	Scattering length	
M	amu	Atomic mass	
c	meV/Å^{-1}	Velocity of sound	
DW	1	Debye-Waller factor	
T	K	Temperature	
target_x	m	position of target to focus at . Transverse coordinate	0
target_y	m	position of target to focus at. Vertical coordinate	0

Name	Unit	Description	Default
target_z	m	position of target to focus at. Straight ahead.	0
target_index	1	relative index of component to focus at, e.g. next is +1	0
focus_r	m	Radius of sphere containing target.	0
focus_xw	m	horiz. dimension of a rectangular area	0
focus_yh	m	vert. dimension of a rectangular area	0
focus_aw	deg	horiz. angular dimension of a rectangular area	0
focus_ah	deg	vert. angular dimension of a rectangular area	0
gap	meV	Bandgap energy (unphysical)	0
e_steps_low	1	Amount of possible intersections beneath the elastic line	50
e_steps_high	1	Amount of possible intersections above the elastic line	50

Links

- Component source code found in file `Phonon_simple.comp`.
- The test/example instrument Test_Phonon.instr.

A simple phonon sample

This component models a simple phonon signal from a single crystal of a pure element in an *fcc* crystal structure. Only one isotropic acoustic phonon branch is modelled, and the longitudinal and transverse dispersions are identical with the velocity of sound being c. Other physical parameters are the atomic mass, M, the lattice parameter, a, the scattering length, b, the Debye-Waller factor, `DW`, and the temperature, T. Incoherent scattering and absorption are taken into account by the cross sections σ_{abs} and σ_{inc}.

The sample can have the form of a cylinder with height h and radius r_0, or a box with dimensions w_x, h_y, t_z.

Phonons are emitted into a specific range of solid angles, specified by the location (x_t, y_t, z_t) and the focusing radius, r_0. Alternatively, the focusing is given by a rectangle, w_{focus} and h_{focus}, and the focus point is given by the index of a down-stream component, `target_index`.

Multiple scattering is not included in this component.

A usage example of this component can be found in the `Neutron site/tests/Test_Phonon` instrument from the `mcgui`.

9.6.1. The phonon cross section

The inelastic phonon cross section for a Bravais crystal of a pure element is given by Ref. [Squ78, ch.3]

$$\begin{aligned}\frac{d^2\sigma'}{d\Omega dE_\mathrm{f}} &= b^2\frac{k_\mathrm{f}}{k_\mathrm{i}}\frac{(2\pi)^3}{V_0}\frac{1}{2M}\exp(-2W)\\ &\times \sum_{\tau,q,p}\frac{(\boldsymbol{\kappa}\cdot\mathbf{e}_{q,p})^2}{\omega_{q,p}}\left\langle n_{q,p}+\frac{1}{2}\mp\frac{1}{2}\right\rangle\delta(\omega\pm\omega_{q,p})\delta(\kappa\pm\mathbf{q}-\tau),\end{aligned}\tag{9.31}$$

where both annihilation and creation of one phonon is considered (represented by the plus and minus sign in the dispersion delta functions, respectively). In the equation, $\exp(-2W)$ is the Debye-Waller factor, DW and V_0 is the volume of the unit cell. The sum runs over the reciprocal lattice vectors, τ, over the polarisation index, p, and the N allowed wave vectors $\mathbf{q}$ within the Brillouin zone (where N is the number of unit cells in the crystal). Further, $\mathbf{e}_{q,p}$ is the polarization unit vectors, $\omega_{q,p}$ the phonon dispersion, and the Bose factor is $\langle n_{q,p}\rangle = (\hbar\exp(|\omega_{q,p}|/k_\mathrm{B}T)-1)^{-1}$.

We have simplified this expression by assuming no polarization dependence of the dispersion, giving $\sum_p(\boldsymbol{\kappa}\cdot\mathbf{e}_{q,p})^2=\kappa^2$. We assume that the inter-atomic interaction is nearest-neighbour-only so that the phonon dispersion becomes:

$$d_1(\mathbf{q}) = c_1/a\sqrt{z-s_q},\tag{9.32}$$

where $z=12$ is the number of nearest neighbours and $s_q=\sum_\mathrm{nn}\cos(\mathbf{q}\cdot\mathbf{r}_\mathrm{nn})$, where in turn $\mathbf{r}_\mathrm{nn}$ is the lattice positions of the nearest neighbours.

This dispersion relation may be modified with a small effort, since it is given as a separate c-function attatched to the component.

To calculate $d\sigma/d\Omega$ we need to transform the $\mathbf{q}$ sum into an integral over the Brillouin zone by $\sum_q\to NV_\mathrm{c}(2\pi)^{-3}\int_\mathrm{BZ}d^3\mathbf{q}$. The $\boldsymbol{\kappa}$ sum can now be removed by expanding the $\mathbf{q}$ integral to infinity. All in all, the partial differential cross section reads

$$\begin{aligned}\frac{d^2\sigma'}{d\Omega dE_\mathrm{f}}(\boldsymbol{\kappa},\omega) &= Nb^2\frac{k_\mathrm{f}}{k_\mathrm{i}}\frac{1}{2M}\int\frac{\hbar\kappa^2}{\hbar\omega_q}\left\langle n_q+\frac{1}{2}\mp\frac{1}{2}\right\rangle\delta(\omega\pm\omega_q)\delta(\boldsymbol{\kappa}\pm\mathbf{q})d^3\mathbf{q}\\ &= Nb^2\frac{k_\mathrm{f}}{k_\mathrm{i}}\frac{\hbar^2\kappa^2}{2M\hbar\omega_q}\left\langle n_\kappa+\frac{1}{2}\pm\frac{1}{2}\right\rangle\delta(\hbar\omega\pm d_1(\kappa)).\end{aligned}\tag{9.33}$$

9.6.2. The algorithm

All neutrons, which hit the sample volume, are scattered into a particular range of solid angle, $\Delta\Omega$, like many other components. One of the difficult things in scattering from a dispersion is to take care to fulfill the dispersion criteria and to find the correct weight transformation.

In **Phonon_simple**, the following steps are taken:

1. If the sample is hit, calculate the total path length inside the sample, otherwise leave the neutron ray unchanged.

2. Choose a scattering point inside the sample
3. Choose a direction for the final wave vector, $\hat{\mathbf{k}}_\mathrm{f}$ within $\Delta\Omega$.
4. Calculate possible values of k_f so that the dispersion relation is fulfilled for the corresponding value of $\mathbf{k}_\mathrm{f}$. (There is always at least one possible k_f value [Bac75].)
5. Choose one of the calculated k_f values.
6. Propagate the neutron to the scattering point and adjust the neutron velocity according to k_f.
7. Calculate and apply the correct weight factor correction, see below.

9.6.3. The weight transformation

Before making the weight transformation, we need to calculate the probability for scattering along one certain direction Ω from one phonon mode. To do this, we must integrate out the delta functions in the cross section (9.33). We here use that $\hbar\omega_q = \hbar^2(k_i^2 - k_f^2)/(2m_\mathrm{N})$, $\kappa = \mathbf{k}_\mathrm{i} - k_\mathrm{f}\hat{\mathbf{k}}_\mathrm{f}$, and the integration rule $\int \delta(f(x)) = (df/dx)(0)^{-1}$. Now, we reach

$$\left(\frac{d\sigma'}{d\Omega}\right)_j = \int \frac{d^2\sigma'}{d\Omega dE_\mathrm{f}} dE_\mathrm{f} = Nb^2 \frac{k_\mathrm{f}}{k_\mathrm{i}} \frac{\hbar^2\kappa^2}{2Md_1(\kappa_j)J(k_{\mathrm{f},j})} \left\langle n_\kappa + \frac{1}{2} \pm \frac{1}{2} \right\rangle . \tag{9.34}$$

where the Jacobian reads

$$J = 1 - \frac{m_\mathrm{N}}{k_\mathrm{f}\hbar^2} \frac{\partial}{\partial k_\mathrm{f}} \left(d_1(\kappa)\right). \tag{9.35}$$

A rough order-of-magnitude consideration gives $\frac{k_{\mathrm{f},j}}{k_\mathrm{i}} \approx 1$, $J \approx 1$, $\langle n_\kappa + \frac{1}{2} \pm \frac{1}{2} \rangle \approx 1$, $\frac{\hbar^2\kappa^2}{2Md_1(\kappa)} \approx \frac{m}{M}$. Hence, $\left(\frac{d\sigma}{d\Omega}\right)_j \approx Nb^2 \frac{m}{M}$, and the phonon cross section becomes a fraction of the total scattering cross section $4\pi Nb^2$, as it must be. The differential cross section per unit volume is found from (9.34) by replacing N with $1/V_0$.

The total weight transformation now becomes

$$\pi_i = a_\mathrm{lin} l_\mathrm{max} n_\mathrm{s} \Delta\Omega b^2 \frac{k_{\mathrm{f},j}}{k_\mathrm{i}} \frac{\hbar^2\kappa}{2V_0 M d_1(\kappa) J(k_{\mathrm{f},j})} \left\langle n_\kappa + \frac{1}{2} \pm \frac{1}{2} \right\rangle , \tag{9.36}$$

where n_s is the number of possible dispersion values in the chosen direction.

The `Test_Phonon` test/example instrument exists in the distribution for this component.

9.7. The `Isotropic_Sqw` McStas Component

Isotropic sample handling multiple scattering and absorption for a general S(q,w) (coherent and/or incoherent/self)

Identification

- **Author:** E. Farhi, V. Hugouvieux
- **Origin:** ILL
- **Date:** August 2003

Description

An isotropic sample handling multiple scattering and including as input the dynamic structure factor of the chosen sample (e.g. from Molecular Dynamics). Handles elastic/inelastic, coherent and incoherent scattering - depending on the input S(q,w) - with multiple scattering and absorption. Only the norm of q is handled (not the vector), and thus suitable for liquids, gazes, amorphous and powder samples.

If incoherent/self S(q,w) file is specified as empty (0 or "") then the scattering is constant isotropic (Vanadium like). In case you only have one S(q,w) data containing both coherent and incoherent contributions you should e.g. use 'Sqw_coh' and set 'sigma_coh' to the total scattering cross section. Set sigma_coh and sigma_inc to -1 to inactivate.

The implementation will automatically nornalise S(q,w) so that S(q) -> 1 at large q (parameter norm=-1). Alternatively, the S(q,w) data will be multiplied by 'norm' for positive values. Use norm=0 or 1 to use the raw data as input.

The material temperature can be defined in the S(q,w) data files (see below) or set manually as parameter T. Setting T=-1 disables detailed balance. Setting T=-2 attempts to guess the temperature from the input S(q,w) data which must then be non-classical and extend on both energy sides (+/-). To use the S(q,w) data as is, without temperature effect, set T=-1 and norm=1.

Both non symmetric (quantum) and classical S(q,w) data sets can be given by mean of the 'classical' parameter (see below).

Additionally, for single order scattering (order=1), you may restrict the vertical spreading of the scattering area using d_phi parameter.

An important option to enhance statistics is to set 'p_interact' to, say, 30 percent (0.3) in order to force a fraction of the beam to scatter. This will result on a larger number of scattered events, retaining intensity.

If you use this component and produce valuable scientific results, please cite authors with references bellow (in Links). E. Farhi et al, J Comp Phys 228 (2009) 5251

Sample shape: Sample shape may be a cylinder, a sphere, a box or any other shape

```
box/plate:       xwidth x yheight x zdepth (thickness=0)
```

hollow box/plate:xwidth x yheight x zdepth and thickness>0

```
cylinder:        radius x yheight (thickness=0)
```

hollow cylinder: radius x yheight and thickness>0

```
sphere:          radius (yheight=0 thickness=0)
hollow sphere:   radius and thickness>0 (yheight=0)
any shape:       geometry=OFF, PLY or STL file
```

The complex geometry option handles any closed non-convex polyhedra. It computes the intersection points of the neutron ray with the object transparently, so that it can be used like a regular sample object. It supports the OFF (incl. NOFF, COFF), PLY and STL file formats. A mesh must be a single, closed object: the neutron path through it is taken from the first entry and exit only, so a second part of the mesh further along the path (disjoint parts, or a concave shape the path re-enters) is not seen. Such files may be generated from XYZ data using: qhull < coordinates.xyz Qx Qv Tv o > geomview.off or powercrust coordinates.xyz and viewed with geomview or java -jar jroff.jar (see below). The default size of the object depends of the OFF file data, but its bounding box may be resized using xwidth,yheight and zdepth.

Concentric components: This component has the ability to contain other components when used in hollow cylinder geometry (namely sample environment, e.g. cryostat and furnace structure). Such component 'shells' should be split into input and output side surrounding the 'inside' components. First part must then use 'concentric=1' flag to enter the inside part. The component itself must be repeated to mark the end of the concentric zone. The number of concentric shells and number of components inside is not limited.

COMPONENT S_in = Isotropic_Sqw(Sqw_coh="Al.laz", concentric=1, ...) AT (0,0,0) RELATIVE sample_position

COMPONENT something_inside ... // e.g. the sample itself or other materials

COMPONENT S_out = COPY(S_in)(concentric=0) AT (0,0,0) RELATIVE sample_position

Sqw file format: File format for S(Q,w) (coherent and incoherent) should contain 3 numerical blocks, defining q axis values (vector), then energy axis values (vector), then a matrix with one line per q axis value, containing Sqw values for each energy axis value. Comments (starting with '#') and non numerical lines are ignored and used to separate blocks. Sampling must be regular. Some parameters can be specified in comment lines, namely (00 is a numerical value):

```
# sigma_abs   00 absorption scattering cross section in [barn]
# sigma_inc   00 coherent scattering cross section in [barn]
# sigma_coh   00 incoherent scattering cross section in [barn]
```

Temperature 00 in [K]

```
# V_rho       00 atom density per Angs^3
# density     00 in [g/cm^3]
```

```
# weight      00 in [g/mol]
# classical   00 [0=contains Bose factor (measurement) ; 1=classical symmetric]
```

Example: # q axis values # vector of m values in Angstroem-1

```
0.001000 .... 3.591000
```

w axis values # vector of n values in meV

```
0.001391 ... 1.681391
```

sqw values (one line per q axis value) # matrix of S(q,w) values (m rows x n values), one line per q value,

```
9.721422  10.599145 ... 0.000000
10.054191 11.025244 ... 0.000000
```

...

```
0.000000             ... 3.860253
```

See for instance file He4_liq_coh.sqw. Such files may be obtained from e.g. INX, Nathan, Lamp and IDA softwares, as well as Molecular Dynamics (nMoldyn). When the provided S(q,w) data is obtained from the classical correlation function G(r,t), which is real and symmetric in time, the 'classical=1' parameter should be set in order to multiply the file data with exp(hw/2kT). Otherwise, the S(q,w) is NOT symmetrised (classical). If the S(q,w) data set includes both negative and positive energy values, setting 'classical=-1' will attempt to guess what type of S(q,w) it is. The temperature can also be determined this way. In case you do not know if the data is classical or quantum, assume it is usually classical at high temperatures, and quantum otherwise (T < typical mode excitations). The positive energy values correspond to Stokes processes, i.e. material gains energy, and neutrons loose energy. The energy range is symmetrized to allow up and down scattering, taking into account detailed balance exp(-hw/2kT).

You may also generate such S(q,w) 2D files using iFit

Powder file format: Files for coherent elastic powder scattering may also be used. Format specification follows the same principle as in the PowderN component, with parameters:

powder_format= Crystallographica: { 4,5,7,0,0,0,0, 0,0 }

```
Fullprof:          { 4,0,8,0,0,5,0, 0,0 }
Undefined:         { 0,0,0,0,0,0,0, 0,0 }
Lazy:              {17,6,0,0,0,0,0,13,0 }
qSq:               {-1,0,0,0,0,0,1, 0,0 }  // special case for [q,Sq] table
or:                {j,d,F2,DW,Delta_d/d,1/2d,q,F,strain}
```

or column indexes (starting from 1) given as comments in the file header (e.g. '#column_j 4'). Refer to the PowderN component for more details. Delta_d/d and Debye-Waller factor may be specified for all lines with the 'powder_Dd' and 'powder_DW' parameters. The reflection list should be ordered by decreasing d-spacing values.

Additionally a special [q,Sq] format is also defined with: powder_format=qSq for which column 1 is 'q' and column 2 is 'S(q)'.

Examples: 1- Vanadium-like incoherent elastic scattering Isotropic_Sqw(radius=0.005, yheight=0.01, V_rho=1/13.827, sigma_abs=5.08, sigma_inc=4.935, sigma_coh=0)

2- liq-4He parameters Isotropic_Sqw(..., Sqw_coh="He4_liq_coh.sqw", T=10, p_interact=0.3)

3- powder sample Isotropic_Sqw(..., Sqw_coh="Al.laz")

%BUGS: When used in concentric mode, multiple bouncing scattering (traversing the hollow part) is not taken into account.

%VALIDATION For Vanadium incoherent scattering mode, V_sample, PowderN, Single_crystal and Isotropic_Sqw produce equivalent results, eventhough the two later are more accurate (geometry, multiple scattering). Isotropic_Sqw gives same powder patterns as PowderN, with an intensity within 20 %.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
powder_format	no quotes	name or definition of column indexes in file	{0,0,0,0,0,0,0,0,0}
Sqw_coh	str	Name of the file containing the values of Q, w and S(Q,w) Coherent part; Q in Å-1, E in meV, S(q,w) in meV-1. Use 0, NULL or "" to disable.	0
Sqw_inc	str	Name of the file containing the values of Q, w and S(Q,w). Incoherent (self) part. Use 0, NULL or "" to scatter isotropically (V-like).	0
geometry	str	Name of an OFF, PLY or STL file for complex geometry. An OFF file may be generated from XYZ coordinates using qhull/powercrust	0
radius	m	Outer radius of sample in (x,z) plane. cylinder/sphere.	0
thickness	m	Thickness of hollow sample Negative value extends the hollow volume outside of the box/cylinder.	0
xwidth	m	width for a box sample shape	0

Name	Unit	Description	Default
yheight	m	Height of sample in vertical direction for box/cylinder shapes	0
zdepth	m	depth for a box sample shape	0
threshold	1	Value under which S(Q,w) is not accounted for. to set according to the S(Q,w) values, i.e. not too low.	1e-20
order	1	Limit multiple scattering up to given order 0:all (default), 1:single, 2:double, ...	0
T	K	Temperature of sample, detailed balance. Use T=-1 to disable it, and T=-2 to guess it from non-classical S(q,w) input.	0
verbose	1	Verbosity level (0:silent, 1:normal, 2:verbose, 3:debug). A verbosity>1 also computes dispersions and S(q,w) analysis.	1
d_phi	deg	scattering vertical angular spreading (usually the height of the next component/detector). Use 0 for full space. This is only relevant for single scattering (order=1).	0
concentric	1	Indicate that this component has a hollow geometry and may contain other components. It should then be duplicated after the inside part (only for box, cylinder, sphere) [1]	0
rho	$Å^{-3}$	Density of scattering elements (nb atoms/unit cell V_0).	0
sigma_abs	barns	Absorption cross-section at 2200 m/s. Use -1 to inactivate.	0
sigma_coh	barns	Coherent Scattering cross-section. Use -1 to inactivate.	0
sigma_inc	barns	Incoherent Scattering cross-section. Use -1 to inactivate.	0
classical	1	Assumes the S(q,w) data from the files is a classical S(q,w), and multiply that data by exp(hw/2kT) on up/down energy sides. Use 0 when obtained from raw experiments, 1 from molecular dynamics. Use -1 to guess from a data set including both energy sides.	-1
powder_Dd	1	global Delta_d/d spreading, or 0 if ideal.	0
powder_DW	1	global Debey-Waller factor, if not in |F2| or 1.	0

Name	Unit	Description	Default
powder_Vc	Å^3	volume of the unit cell	0
density	g/cm^3	density of material. V_rho=density/weight/1e24*N_A	0
weight	g/mol	atomic/molecular weight of material	0
p_interact	1	Force a given fraction of the beam to scatter, keeping intensity right, to enhance small signals (-1 inactivate).	-1
norm	1	Normalize S(q,w) when -1 (default). Use raw data when 1, multiplier for S(q,w) when norm>0.	-1
powder_barns	1	0 when |F2| data in powder file are fm^2, 1 when in barns (barns=1 for laz, barns=0 for lau type files).	1

Links

- Component source code found in file `Isotropic_Sqw.comp`.
- E. Farhi, V. Hugouvieux, M.R. Johnson, and W. Kob, Journal of Computational Physics 228 (2009) 5251-5261 "Virtual experiments: Combining realistic neutron scattering instrument and sample simulations"
- Hugouvieux V, Farhi E, Johnson MR, Physica B, 350 (2004) 151 "Virtual neutron scattering experiments"
- Hugouvieux V, PhD, University of Montpellier II, France (2004).
- H. Schober, Collection SFN 10 (2010) 159-336
- H.E. Fischer, A.C. Barnes, and P.S. Salmon. Rep. Prog. Phys., 69 (2006) 233
- P.A. Egelstaff, An Introduction to the Liquid State, 2nd Ed., Oxford Science Pub., Clarendon Press (1992).
- S. W. Lovesey, Theory of Neutron Scattering from Condensed Matter, Vol1, Oxford Science Pub., Clarendon Press (1984).
- Cross sections for single elements
- Cross sections for compounds
- Web Elements
- Fullprof powder refinement
- Crystallographica software

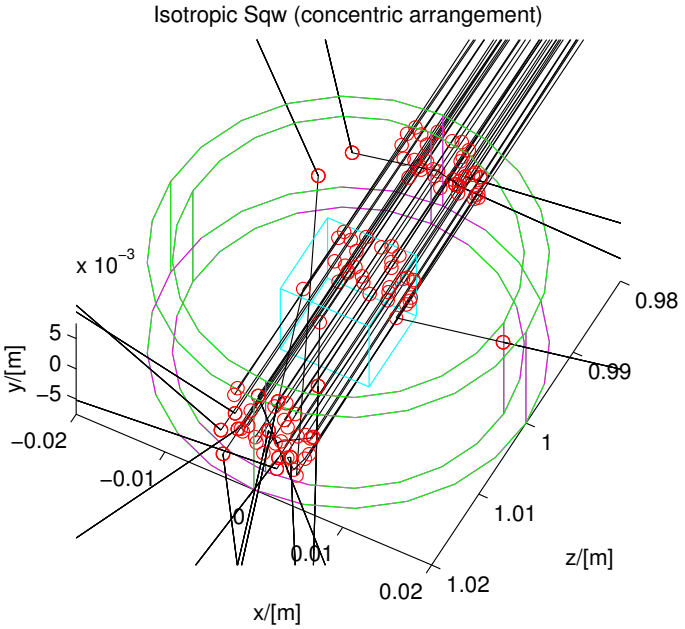


Figure 9.6.: An $l-^4$He sample in a cryostat, simulated with the Isotropic_Sqw component in concentric geometry.

- Example data file He4_liq_coh.sqw
- The PowderN component.
- The test/example instrument Test_Isotropic_Sqw.instr.
- Geomview and Object File Format (OFF)
- Java version of Geomview (display only) jroff.jar
- qhull
- powercrust

A general $S(q,\omega)$ coherent and incoherent scatterer

The sample component *Isotropic_Sqw* has been developed in order to simulate neutron scattering from any isotropic material such as liquids, glasses (amorphous systems), polymers and powders (currently, mono-crystals cannot be handled). The component treats coherent and incoherent neutron scattering and may be used to model most materials,

including sample environments with concentric geometries. The structure and dynamics of isotropic samples can be characterised by the dynamic structure factor $S(q,\omega)$, which determines the interaction between neutrons and the sample and therefore can be used as a probability distribution of ω-energy and q-momentum transfers. It handles coherent and incoherent processes, both for elastic and inelastic interactions. The main input for the component is $S(q,\omega)$ tables, or powder structure files.

Usage examples of this component can be found in the
`Neutron site/tests/Test_Isotropic_Sqw`, the
`Neutron site/ILL/ILL_H15_IN6` and the `ILL_TOF_Env` instruments from the `mcgui`.

From McStas 2.4 we have decided to include the earlier version of Isotropic_Sqw from the McStas 2.0 under the name of Isotropic_Sqw_legacy, since some users reported that release being in better agreement with experiments. Note however that issues were corrected since 2.0 and fixed in today's component, that on the other hand exhibits other issues in terms of multiple scattering etc. We will try to rectify the problems during 2017.

9.7.1. Neutron interaction with matter - overview

When a neutron enters a material, according to usual models, it 'sees' atoms as disks with a surface equal to the total cross section of the material σ_{tot}. The latter includes absorption, coherent and incoherent contributions, which all depend on the incoming neutron energy. The transmission probability follows an exponential decay law accounting for the total cross section.

For the neutron which is not transmitted, we select a scattering position along the path, taking into account the secondary extinction and absorption probability. In this process, the neutron is considered to be a particle or an attenuated wave.

Once a scattering position has been assigned, the neutron interacts with a material excitation. Here we turn to the wave description of the neutron, which interacts with the whole sample volume. The distribution of excitations, which determines their relative intensity in the scattered beam, is simply the dynamic structure factor - or scattering law - $S(q,\omega)$. We shall build probability distributions from the scattering law in order to improve the efficiency of the method by favoring the (q,ω) choice towards high $S(q,\omega)$ regions.

The neutron leaves the scattering point when a suitable (q,ω) choice has been found to satisfy the conservation laws. The method is iterated until the neutron leaves the volume of the material, therefore allowing multiple scattering contributions, which will be considered in more details below.

No experimental method makes it possible to accurately measure the multiple scattering contribution, even though it can become significant at low q transfers (below the first diffraction maximum), where the single scattering coherent signal is weak in most materials. This is why attemps have been made to reduce the multiple scattering contribution by partitioning the sample with absorbing layers. However, this is not always applicable thus makiong the simulation approach very valuable.

The method presented here for handling neutron interaction with isotropic materials is similar in many respects to the earlier MSC [FMW72], Discus [Johrc] and MSCAT

[Cop74] methods, but the implementation presented here is part of a more general treatment of a sample in an instrument.

9.7.2. Theoretical side

Pair correlation function $g(r)$ and Dynamic structure factor $S(q,\omega)$

In the following, we consider an isotropic medium irradiated with a cold or thermal neutron beam. We ignore the possible thermal fission events and assume that the incoming neutron energy does not correspond to a Breit-Wigner resonance in the material. Furthermore, we do not take into account quantum effects in the material, nor refraction and primary extinction.

Following Squires [Squ78], the experimental counterpart of the scattering law $S(q,\omega)$ is the neutron double differential scattering cross section for both coherent and incoherent processes:

$$\frac{d^2\sigma}{d\Omega dE_f} = \frac{\sigma}{4\pi}\frac{k_f}{k_i}NS(q,\omega) \tag{9.37}$$

which describes the amount of neutrons scattered per unit solid angle $d\Omega$ and per unit final energy dE_f. In this equation, $N = \rho V$ is the number of atoms in the scattering volume V with atomic number density ρ, E_f, E_i, k_f, k_i are the kinetic energy and wavevectors of final and initial states respectively, σ is the bound atom scattering cross-section, Ω is the solid angle and q, ω are the wave-vector and energy transfer at the sample. In practice, the double differential cross section is a linear combinaison of the coherent and incoherent parts as:

$$\sigma S(q,\omega) = \sigma_{coh}S_{coh}(q,\omega) + \sigma_{inc}S_{inc}(q,\omega) \tag{9.38}$$

where the subscripts *coh* and *inc* stand for the coherent and incoherent contributions respectively.

We define its norm on a selected q range:

$$|S| = \iint S(q,\omega)dqd\omega. \tag{9.39}$$

The norm $\lim_{q\to\infty}|S| \simeq q$ for large q values, and can only be defined on a restricted q range.

Some easily measureable coherent quantities in a liquid are the *static pair correlation function* $g(r)$ and the *structure factor* $S(q)$, defined as:

$$\begin{aligned}
\rho g(\vec{r}) &= \frac{1}{N}\sum_{i=1}^{N}\sum_{j\neq i}\langle\delta(\vec{r}+\vec{r}_i-\vec{r}_j)\rangle \\
S(\vec{q}) &= \int S(\vec{q},\omega)d\omega \\
&= 1+\rho\int_V [g(\vec{r})-1]e^{i\vec{q}.\vec{r}}d\vec{r} \\
&= 1+\rho\int_0^\infty [g(r)-1]\frac{\sin(qr)}{qr}4\pi r^2 dr \text{ in isotropic materials.}
\end{aligned} \tag{9.40}$$

The latter expression, in isotropic materials, may be Fourier transformed as:

$$g(r) - 1 = \frac{1}{2\pi^2 \rho} \int_0^\infty q^2 [S(q) - 1] \frac{sin(qr)}{qr} dq \tag{9.41}$$

Both $g(r)$ and $S(q)$ converge to unity for large r and q values respectively, and they are representative of the atoms spatial distribution. In a liquid $\lim_{q\to 0} S(q) = \rho k_B T \chi_T$ where $\chi_T = (\frac{\partial \rho}{\partial P})_{V,T}$ is the compressibility [Ege67; FBS06]. In perfect gases, $S(q) = 1$ for all q. These quantities are obtained experimentally from diffractometers. In principle, $S_{inc}(q) = 1$ in all materials, but a q dependence is rather usual, partly due to the Debye-Waller factor $e^{-q^2\langle u^2 \rangle}$. Anyway, $S_{inc}(q)$ converges to unity at high q.

The static pair correlation function $g(r)$ is the probability to find a neighbouring atom at a given distance (unitless). Since $g(0) = 0$, Eq. (9.41) provides a useful normalisation sum-rule for coherent $S(q)$:

$$\int_0^\infty q^2 [S(q) - 1] dq = -2\pi^2 \rho \text{ for coherent contribution.} \tag{9.42}$$

This means that the integrated oscillations (around 1) of $S_{coh}(q)$ are directly related to the density of the material ρ. In practice, the function $S(q)$ is often known on a restricted range $q \in [0, q_{max}]$, due to either limitations in the sample molecular dynamics simulation, or the measurement itself. In first approximation we consider that Eq. (9.42) can be applied in this range, i.e. we neglect the large q contributions provided $S(q) - 1$ converges faster than $1/q^2$. This is usually true after 2-3 oscillations of $S(q)$ in liquids. Then, in isotropic liquid-like materials, Eq. (9.42) provides a normalisation sum-rule for S.

9.7.3. Theoretical side - scattering in the sample

The Eq. 9.37 controls the scattering in the whole sample volume. Its implementation in a propagative Monte Carlo neutron code such as *McStas* can be summarised as follows:

1. Compute the propagation path length in the material by geometrical intersections between the neutron trajectory and the sample volume.

2. Evaluate the total cross section from the integration of the scattering law over the accessible dynamical range (Section 9.7.3).

3. Use the total cross section to determine the probability of interaction for each neutron along the path length, and select a scattering position.

4. Weight neutron interaction with the absorption probability and select the type of interaction (coherent or incoherent).

5. Select the wave vector and energy transfer from the dynamic structure factor $S(q, \omega)$ used as a probability distribution (Section 9.7.3). Apply the detailed balance.

6. Check whether selection rules can be solved (Section 9.7.3). If they cannot, repeat (5).

This procedure is iterated until the neutron leaves the sample. We shall now detail the key steps of this implementation.

Evaluating the cross sections and interaction probability

Following Sears [Sea75], the total scattering cross section for incoming neutrons with initial energy E_i is

$$\sigma_s(E_i) = \iint \frac{d^2\sigma}{d\Omega dE_f} d\Omega dE_f = \frac{N\sigma}{4\pi} \iint \frac{k_f}{k_i} S(q,\omega) d\Omega dE_f \tag{9.43}$$

where the integration runs over the entire space and all final neutron energies. As the dynamic structure factor is defined in the q, ω space, the integration requires a variable change. Using the momentum conservation law and the solid angle relation $\Omega = 2\pi(1 - cos\theta)$, were θ is the solid angle opening, we draw:

$$\sigma_s(E_i) = N \iint \frac{\sigma S(q,\omega) q}{2k_i^2} dq d\omega. \tag{9.44}$$

This integration runs over the whole accessible q, ω dynamical range for each incoming neutron. In practice, the knowledge of the dynamic structure factor is defined over a limited area with $q \in [q_{min}, q_{max}]$ and $\omega \in [\omega_{min}, \omega_{max}]$ which is constrained by the method for obtaining $S(q,\omega)$, i.e. from previous experiments, molecular dynamics simulations, and analytical models. It is desirable that this area be as large as possible, starting from 0 for both ranges. If we use $\omega_{min} \to 0$, $q_{min} \to 0$, $\omega_{max} > 4E_i$ and $q_{max} > 2k_i$, we completely describe all scattering processes for incoming neutrons with wavevector k_i [FMW72].

This means that in order to correctly estimate the total intensity and multiple scattering, the knowledge of $S(q,\omega)$ must be wider (at least twice in q, as stated previously) than the measurable range in the corresponding experiment. As a side effect, a self consistent iterative method for finding the true scattering law from the measurement itself is not theorically feasible, except for providing crude approximations. However, that measured dynamic structure factor may be used to estimate the multiple scattering for a further measurement using longer wavelength neutrons. In that case, extrapolating the scattering law beyond the accessible measurement ranges might improve substantially the accuracy of the method, but this discussion is beyond the scope of this paper.

Consequently, limiting the q integration in Eq. 9.44 to the maximum momentum transfer for elastic processes $2k_i$, we write the total scattering cross section as

$$\sigma_s(E_i) \simeq \frac{N}{2k_i^2} \int_0^{2k_i} q\sigma S(q) dq. \tag{9.45}$$

Using Eq. 9.38, it is possible to define similar expressions for the coherent and incoherent terms $\sigma_{coh}(E_i)$ and $\sigma_{inc}(E_i)$ respectively. These integrated cross sections are usually

quite different from the tabulated values [DL03] since the latter are bound scattering cross sections.

Except for a few materials with absorption resonances in the cold-thermal energy range, the absorption cross section for an incoming neutron of velocity $v_i = \sqrt{2E_i/m}$, where m is the neutron mass, is computed as $\sigma_{abs}(E_i) = \sigma_{abs}^{2200} \frac{2200m/s}{\sqrt{2E_i/m}}$, where σ_{abs}^{2200} is obtained from the literature [DL03].

We now determine the total cross section accounting for both scattering and absorption

$$\sigma_{tot}(E_i) = \sigma_{abs}(E_i) + \sigma_s(Ei). \tag{9.46}$$

The neutron trajectory intersection with the sample geometry provides the total path length in the sample d_{exit} to the exit. Defining the linear attenuation $\mu(E_i) = \rho\sigma_{tot}(E_i)$, the probability that the neutron event is transmitted along path d_{exit} is $e^{-\mu(E_i)d_{exit}}$.

If the neutron event is transmitted, it leaves the sample. In previous Monte Carlo codes such as DISCUSS [Johrc], MSC [FMW72] and MSCAT [Cop74], each neutron event is forced to scatter to the detector area in order to improve the sample scattering simulation statistics and reduce the computing time. The corresponding instrument model is limited to a neutron event source, a sample and a detector. It is equaly possible in the current implementation to 'force' neutron events to scatter by applying a correction factor $\pi_0 = 1 - e^{-\mu(E_i)d_{exit}}$ to the neutron statistical weight. However, the *McStas* instrument model is often build from a large sequence of components. Eventhough the instrument description starts as well with a neutron event source, more than one sample may be encountered in the course of the neutron propagation and multiple detectors may be positioned anywhere in space, as well as other instrument components (e.g. neutron optics). This implies that neutron events scattered from a sample volume should not focus to a single area. Indeed, transmitted events may reach other scattering materials and it is not desirable to force all neutron events to scatter. The correction factor π_0 is then not applied, and neutron events can be transmitted through the sample volume. The simulation efficiency for the scattering then drops significantly, but enables to model much more complex arrangements such as concentric sample environments, magnets and monochromator mechanical parts, and neutron filters.

If the neutron is not transmitted, the neutron statistical weight is multiplied by a factor

$$\pi_1 = \frac{\sigma_s(E_i)}{\sigma_{tot}(E_i)} \tag{9.47}$$

to account for the fraction of absorbed neutrons along the path, and we may in the following treat the event as a scattering event. Additionally, the type of interaction (coherent or incoherent) is chosen randomly with fractions $\sigma_{coh}(E_i)$ and $\sigma_{inc}(E_i)$.

The position of the neutron scattering event along the neutron trajectory length d_{exit} is determined by [MPC77; Johrc]

$$d_s = -\frac{1}{\mu(E_i)} \ln(1 - \xi[1 - e^{-\mu(E_i)d_{exit}}]) \tag{9.48}$$

where ξ is a random number in [0,1]. This expression takes into account secondary extinction, originating from the decrease of the beam intensity through the sample (self shielding).

Choosing the q and ω transfer from $S(q,\omega)$

The choice of the (q,ω) wavevector-energy transfer pair could be done randomly, as in the first event of the second order scattering evaluation in DISCUS [Johrc], but it is somewhat inefficient except for materials showing a broad quasi-elastic signal. As the scattering originates from structural peaks and excitations in the material $S(q,\omega)$, it is usual [Cop74] to adopt an importance sampling scheme by focusing the (q,ω) choice to areas where the intensity of $S(q,\omega)$ is high. In practice, this means that the neutron event should scatter preferably on e.g. Bragg peaks, quasielastic contribution and phonons.

The main idea to implement the scattering from $S(q,\omega)$ is to cast two consecutive Monte Carlo choices, using probability distribution built from the dynamic structure factor. We define first the probability $P_\omega(\omega)$ as the *unweighted* fraction of modes whose energy lies between ω and $\omega + d\omega$

$$P_\omega(\omega)d\omega = \frac{\int_0^{q_{max}} qS(q,\omega)dq}{|S|}, \tag{9.49}$$

where $|S| = \iint S(q,\omega)dqd\omega$ is the norm of $S(q,\omega)$ in the available dynamical range $q \in [q_{min}, q_{max}]$ and $\omega \in [\omega_{min}, \omega_{max}]$. The probability $P_\omega(\omega)$ is normalised to unity, $\int P_\omega(\omega)d\omega = 1$, and is a probability distribution of mode energies in the material. We then choose randomly an energy transfer ω from this distribution.

Similarly, in order to focus the wavevector transfer choice, we define the probability distribution of wavevector $P_q(q \mid \omega)$ for the selected energy transfer lying between ω and $\omega + d\omega$

$$P_q(q \mid \omega) = \frac{qS(q,\omega)}{S(q)}, \tag{9.50}$$

from which we choose randomly a wavevector transfer q, knowing the energy transfer ω. These two probability distributions extracted from $S(q,\omega)$ are shown in Fig. 9.7, for a model $S(q,\omega)$ function built from the l-^{4}He elementary excitation (Data from Donnelly).

Then a selection between energy gain and loss is performed with the detailed balance ratio $e^{-\hbar\omega/k_BT}$. In the case of Stokes processes, the neutron can not loose more than its own energy to the sample dynamics, so that $\hbar\omega < E_i$. This condition breaks the symmetry between up-scattering and down-scattering.

Solving selection rules and choosing the scattered wave vector

The next step is to check that the conservation laws

$$\hbar\omega = E_i - E_f = \frac{\hbar^2}{2m}(k_i^2 - k_f^2) \tag{9.51}$$

$$\vec{q} = \vec{k}_i - \vec{k}_f \tag{9.52}$$

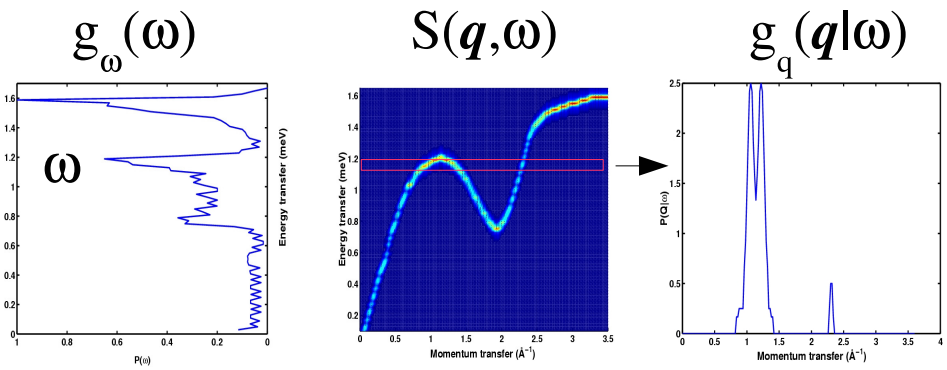


Figure 9.7.: *Centre*: Model of dynamic structure factor $S(q,\omega)$ for l-^{4}He ; *left*: probability distribution g_ω (horizontal axis) of energy transfers (vertical axis, density of states) ; *right* : probability distribution $g_q(\omega)$ (vertical axis) of momentum transfers (horizontal axis) for a given energy transfer $\hbar\omega \sim 1.1$ meV.

can be satisfied. These conditions are closely related to the method for selecting the outgoing wave vector direction.

When the final wave vector has to be computed, the quantities $\vec{k}_i$, $\hbar\omega$ and $q = |\vec{q}|$ are known. We solve the energy conservation law Eq. (9.51) and we select randomly k_f as one of the two roots.

The scattering angle θ from the initial k_i direction is determined from the momentum conservation law $cos(\theta) = (k_i^2 + k_f^2 - q^2)/(2k_ik_f)$, which defines a scattering cone. We then choose randomly a direction on the cone.

If the selection rules can not be verified (namely $|cos(\theta)| > 1$), a new (q,ω) random choice is performed (see Section 9.7.3). It might appear inefficient to select the energy and momentum tranfers first and check the selection rules afterwards. However, in practice, the number of iterations to actually scatter on a high probability process and satisfy these rules is limited, usually below 10. Moreover, as these two steps are simple, the whole process requires a limited number of computer operations.

As mentioned in Section 9.7.3, previous multiple scattering estimation codes [FMW72; Cop74; Johrc] force the outgoing neutron event to come into the detector area and time window, thus improving dramatically the code efficiency. This choice sets the measurable energy and momentum transfers for the last scattering event in the sample, so that the choice of the scattering excitation actually requires a more complex sampling mechanism for the dynamic structure factor. As the present implementation makes no assumption on the simulated instrument part which is behind the sample, we can not apply this method. Consequently, the efficiency of the sample scattering code is certainly lower than previous codes, but on the other hand it does not depend on the type of instrument simulation. In particular, it may be used to model any material in the course of the neutron propagation along the instrument model (filters, mechanical parts, samples,

shields, radiation protections).

Once the scattering probability and position, the energy and momentum transfers and the neutron momentum after scattering have all been defined, the whole process is iterated until the neutron is transmitted and exits the sample volume.

Extension to powder elastic scattering

In principle, the component can work in purely elastic mode if only the $\omega = 0$ column is available in S. Anyway, in the diffractionists world, people do not usually define scattering with $S(q)$ (Eq. 9.40), but through the scattering vector $\boldsymbol{\tau}$, multiplicity $z(\tau)$ (for powders), and $|F^2|$ structure factors including Debye-Waller factors, as in Eq. 9.18.

When doing diffraction, and neglecting inelastic contribution as first approximation, we may integrate Eq. 9.37, keeping $k_i = k_f$.

$$\left(\frac{d\sigma}{d\Omega}\right)_{\text{coh.el.}}(|q|) = \int_0^\infty \frac{d^2\sigma_{coh}}{d\Omega dE_f} dE_f = \frac{N\sigma_{coh}}{4\pi} S_{coh}(q) \tag{9.53}$$

$$= N\frac{(2\pi)^3}{V_0}\sum_{\boldsymbol{\tau}} \delta(\boldsymbol{\tau} - \boldsymbol{q})|F_{\boldsymbol{\tau}}|^2 \text{ from Eq. (9.18)} \tag{9.54}$$

with $V_0 = 1/\rho$ being the volume of a lattice unit cell. Then we come to the formal equivalence, in the powder case [Squ78] (integration over Debye-Scherrer cones):

$$S_{coh}(q) = \frac{\pi\rho}{2\sigma_{coh}}\frac{z(q)}{q^2}|F_q|^2 \text{ in a powder.} \tag{9.55}$$

for each lattice Bragg peak wave vector q. The normalisation rule Eq. (9.42) can not usually be applied for powders, as the $S(q)$ is a set of Dirac peaks for which the $\int q^2 S(q)dq$ is difficult to compute, and $S(q)$ does not converge to unity for large q. Each F^2 Dirac contribution may be broaden when specifiying a diffraction peak width.

Of course, the component PowderN (see section 9.3) can handle powder samples efficiently (faster, better accuracy), but does not take into account multiple scattering, nor secondary extinction (which is significant for materials with large absorption cross sections). On the other side, the current Isotropic_Sqw component assumes a powder packing factor of 1 (massive sample). To change into a lower packing factor, use a lower powder density.

Important remarks and limitations

Since the choice of the interaction type, we know that the neutron *must* scatter, with an appropriate $\vec{k}_f$ outgoing wave vector. If any of the choices in the method fails:

1. the two roots k_f^+ and k_f^- are imaginary, which means that conservation laws can not be satisfied and for instance the selected energy transfer is higher than the incoming neutron energy

2. the radius of the target circle is imaginary, that is $|cos(\theta)| > 1$.

then a new (q, ω) set is drawn, and the process is iterated until success or - at last - removal of the neutron event. These latter absorptions are then reported at the end of the simulation, as it never occurs in reality - neutrons that scatter do find a suitable (q, ω) set.

The $S(q, \omega)$ data sets should be as wide a possible in q and ω range, else scattering conditions will be limited by the reduced data set (specially multiple scattering estimates). On the other hand, when q and ω ranges are too large, some Monte Carlo choices lead to scattering temptatives in non useful regions of S, which reduces dramatically the algorithm efficiency.

The best settings are:

1. to have the widest q and ω range for $S(q, \omega)$ data sets,
2. to either set $wmax$ and $qmax$ to the maximum scatterable energy and wavevectors,
3. or alternatively request the automatic range optimisation by setting parameter `auto_qw=1`. This is recommended, but may sometimes miss a few neutrons if the q, ω beam range has been guessed too small.

Focusing the q and ω range (e.g. with 'auto_qw=1'), to the one being able to scatter the incoming beam, when using the component does improve significantly the speed of the computation. Additionally, if you restrict the scattering to the first order only (parameter 'order=1'), then you may specify the angular vertical extension $d\phi$ of the scattering area to gain optimised focusing. This option does not apply when handling multiple scattering (which emits in 4π many times before exiting the sample).

A bilinear interpolation for the q, ω determination is used to improve the accuracy on the scattered intensity, but it may be unactivated when setting parameter `interpolate=0`. This will often result in a discrete q, ω sampling.

As indicated in the previous section, the Isotropic_Sqw component is not as efficient as PowderN for powder single scattering, but handles scattering processes in a more accurate way (secondary extinction, multiple scattering).

9.7.4. The implementation

Geometry

The geometry for the component may be box, cylinder and sphere shaped, either filled or hollow. Relevant parameters for this purpose are as follow:

- **box**: dimensions are $x_{width} \times y_{height} \times z_{depth}$.
- **box, hollow**: *idem*, and the side wall thickness is set with *thickness*.
- **cylinder**: dimensions are r for the radius and y_{height} for the height.
- **cylinder, hollow**: *idem*, and hollow part is set with *thickness*.
- **sphere**: dimension is r for the radius.

Parameter	type	meaning
Sqw_coh	string	Coherent scattering data file name. Use 0, NULL or "" to disable
Sqw_inc	string	Incoherent scattering data file name. Use 0, NULL or "" to scatter isotropically (Vanadium like)
sigma_coh	[barns]	Coherent scattering cross-section. -1 to disable
sigma_inc	[barns]	Incoherent scattering cross-section. -1 to disable
sigma_abs	[barns]	Absorption cross-section. -1 to disable
V_rho	[Å^{-3}]	atomic number density (component parameter *rho*). May also be specified with molar weight *weight* in [g/mol] and material *density* in [g/cm^3]
T	[K]	Temperature. 0 disables detailed balance
xwidth	[m]	
yheight	[m]	dimensions of a box shaped geometry
zdepth	[m]	
radius	[m]	dimensions of a cylinder or sphere shaped geometry (outer radius)
thickness	[m]	thickness of hollow shape
auto_qw	boolean	Automatically optimise probability tables during simulation
auto_norm	scalar	Normalize $S(q,\omega)$ when -1, use raw data when 0, multiply S by given value when positive
order	integer	Limit multiple scattering up to given order. 0 means all orders
concentric	boolean	Enables to 'enter' inside concentric hollow geometries

Table 9.9.: Main Isotropic_Sqw component parameters

- **sphere, hollow**: *idem*, and hollow part is set with *thickness*.

The AT position corresponds to the centre of the sample.

Hollow shapes are particularly useful to model complex sample environments. Refer to the dedicated section below for more details on this topic.

Dynamical structure factor

The material behaviour is specified through the total scattering cross-sections σ_{coh}, σ_{inc}, σ_{abs}, and the $S(q,\omega)$ data files.

If you are lucky enough to have access to separated coherent and incoherent contributions (e.g. from material simulation), simply set Sqw_coh and Sqw_inc parameter to the files names. If on the other hand you have access to a global data set containing incoherent scattering as well (e.g. the result of a previous experiment), use Sqw_coh parameter, set the σ_{coh} parameter to the sum of both contributions $\sigma_{coh} + \sigma_{inc}$, and set $\sigma_{inc} = -1$. This way we only use one of the two implemented scattering channels. Such global data sets may originate from previous experiments, as far as you have applied all known corrections (multiple scattering, geometry, ...).

In any case, the accuracy of the $S(q,\omega)$ data limits the q and ω resolution of the simulation, eventhough a bilinear interpolation is performed in order to smooth binning. The sampling of data files should then be as thin as possible.

If the Sqw_inc parameter is left unset but the σ_{inc} is *not* zero, an isotropic incoherent elastic scattering is used, just like the V_sample component (see section 9.1).

Anyway, as explained below, it is also possible to simulate the elastic scattering from a powder file (see below).

File formats: $S(q,\omega)$ inelastic scattering

The format of the data files is free text, consisting of three numerical blocks, separated by empty lines or comments, in the following order

1. A vector of length m containing wavevector q values, in Å^{-1}.
2. A vector of length n containing energy ω values, in meV.
3. A matrix of size m rows by n columns, of $S(q,\omega)$ values, in meV^{-1}.

Any line beginning with any character of `#;/%` is considered to be a comment, and lines which can not be read as vectors/matrices are ignored.

The file header may optionally contain parameter settings for the material, as comments, with keywords as in the following example:

```
#V_0          35    cell volume [Angs^3]
#V_rho        0.07  atom number density [at/Angs^3]
#sigma_abs    5     absorption cross section [barns]
#sigma_inc    4.8   incoherent cross section [barns]
#sigma_coh    1     coherent cross section  [barns]
#Temperature  10    for detailed balance [K]
```

```
#density      1     material density [g/cm^3]
#weight       18    material molar weight [g/mol]
#nb_atoms     6     number of atoms per unit cell
```

Some `sqw` data files are included in the McStas distribution data directory, and they contain material parameter settings in their header, so that you may use:

```
Isotropic_Sqw(<geometry parameters>, Sqw_coh="He4_liq_coh.sqw", T=4)
```

Example files are listed as `*.sqw` files in directory `MCSTAS/data`. A table of $S(q,\omega)$ data files for a few liquids are listed in Table 1.3 (page 22).

File formats: $S(q)$ liquids

This file format provides a mean to import directly an $S(q)$ data set, when setting parameters:

```
powder_format=qSq
```

The 'Sqw_coh' (or 'Sqw_inc') file should contains a single numerical block, which column assignment is defaulted as q and $S(q)$ being the first and second column respectively. This may be overridden from the file header with '#column' keywords, as in the example:

```
#column_q  2
#column_Sq 1
```

Such files can only handle elastic scattering.

File formats: powder structures (LAZY, Fullprof, Crystallographica)

Data files as used by the component PowderN may also be read. Data files of type `lau` and `laz` in the McStas distribution data directory are self-documented in their header. They do not need any additional parameters to be used, as in the example:

```
Isotropic_Sqw(<geometry parameters>, Sqw_coh="Al.laz")
```

Other column-based file formats may also be imported e.g. with parameters such as:

```
powder_format=Crystallographica
powder_format=Fullprof
powder_Dd     =0
powder_DW     =1
```

The last two parameters may as well be specified in the data file header with lines:

```
#Debye_Waller 1
#Delta_d/d    1e-3
```

The powder description is then translated into $S(q)$ by using Eq. (9.55). In this case, the density $\rho = n/V_0$ is the number of atoms in the inverse volume of the unit cell.

As the component builds an $S(q)$ from the powder structure description, the accuracy of the Isotropic_Sqw component is limited by the binning during that conversion. This is usually enough to describe sample environments including powders (aluminium, copper, ...), but it is recommended to rather use PowderN for faster and accurate powder diffraction, eventthough this latter does not implement multiple scattering.

Such files can only handle elastic scattering. A list of common powder definition files is available in Table 1.2 (page 24).

Concentric geometries, sample environment

The component has been designed in a way which enables to describe complex imbricated set-ups, i.e. what you need to simulate sample environments. To do so, one has first to use hollow shapes, then keep in mind that each surrounding geometry should be first declared before the central position (usually the sample) with the `concentric=1` parameter, but also duplicated (with an other instance name) at a symmetric position with regards to the centre as in the example (shown in Fig. 9.6):

```
COMPONENT s_in=Isotropic_Sqw(
  thickness=0.001, radius=0.02, yheight=0.015,
  Sqw_coh="Al.laz", concentric=1)
AT (0,0,1) RELATIVE a

COMPONENT sample=Isotropic_Sqw(
  xwidth=0.01, yheight=0.01, zdepth=0.01,
  Sqw_coh="Rb_liq_coh.sqw")
AT (0,0,1) RELATIVE a

COMPONENT s_out=Isotropic_Sqw(
  thickness=0.001, radius=0.02, yheight=0.015,
  Sqw_coh="Al.laz")
AT (0,0,1) RELATIVE a
```

Central component may be of any type, not specifically an Isotropic_Sqw instance. It could be for instance a Single_crystal or a PowderN. In principle, the number of surrounding shells is not restricted. The only restriction is that neutrons that scatter (in 4π) can not come back in the instrument description, so that some of the multiple scattering events are lost. Namely, in the previous example, neutrons scattered by the outer wall of the cryostat `s_out` can not come back to the sample or to the other cryostat wall `s_in`. As these neutrons have usually few chances to reach the rest of the simulation, we expect that the approximation is fair.

9.7.5. Validation

For constant incoherent scattering mode, V_sample, PowderN, Single_crystal and Isotropic_Sqw produce equivalent results, eventhough the two later are more accurate (geometry, multiple scattering). Execution times are equivalent.

Compared with the PowderN component, the $S(q)$ method is twice slower in computation time, and intensity is usually lower by typically 20 % (depending on scattering

cross sections), the difference arising from multiple scattering and secondary extinction (not handled in PowderN). The PowderN component is intrinsically more accurate in q as each Bragg peak is handled separately as an exact Dirac peak, with optional Δq spreading. In Isotropic_Sqw, an approximated $S(q)$ table is built from the F^2 data, and is coarser. Still, differences in the diffraction pattern are limited.

The Isotropic_Sqw component has been benchmarked against real experiment for liquid Rubidium (Copley, 1974) and liquid Cesium (Bodensteiner and Dorner, 1989), and the agreement is excellent.

The `Test_Isotropic_Sqw` test/example instrument exists in the distribution for this component.

9.8. Magnetic samples, magnons and NCrystal

9.9. The `Magnon_bcc` McStas Component

A sample for AFM or FM magnon scattering based on cross section expressions from Squires, Ch.8.2

Identification

- **Author:** Kim Lefmann
- **Origin:** KU
- **Date:** 23.10.08 - 24.07.18

Description

Single-cylinder shape. Absorption included. No multiple scattering. No incoherent scattering emitted. No attenuation from coherent scattering. No Bragg scattering. bcc crystal n.n. and n.n.n. interactions only Can do either FM or AFM order upon a flag Assume J>0 for both FM and AFM. MUST BE CHANGED FOR CONSISTENCY If AFM, the order is two-sublattice, e.g. the AFM Bragg ordering vectors are Q = (1 0 0) and equivalent. One magnon branch only Assume spin along z Possible easy axis anisotropy along z No external field

KNOWN BUGS: Gives zero scattering for too large J values (for AFM J=0.362, h approx 1). Probably this is a malfunction of zridd or call thereof The value of the absolute scattered intensity is clearly too high. This is probably due to unit confusion. The relative intensity scaling seems about right.

Algorithm: 0. Always perform the scattering if possible (otherwise ABSORB) 1. Choose direction within a focusing solid angle 2. Calculate the zeros of (E_i-E_f-hbar omega(kappa)) as a function of k_f 3. Choose one value of k_f (always at least one is possible!) 4. Perform the correct weight transformation

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	m	Outer radius of sample in (x,z) plane	
yheight	m	Height of sample in y direction	
sigma_abs	barns	Absorption cross section at 2200 m/s per atom	
sigma_inc	barns	Incoherent scattering cross section per atom	

Name	Unit	Description	Default
a	Å	bcc Lattice constant (r0: [fm] Classical electron radius) (gamma: [1] Neutron gyromagnetic moment)	
FM	1	Flag for whether the order if FM (0 means AFM)	0
J1	meV	spin-spin interaction 1 (nn)	
J2	meV	spin-spin interaction 2 (nnn)	
D	mev	single ion anisotropy	
s	1	spin	
DW	1	Debye-Waller factor	
T	K	Temperature	
target_x	m	position of target to focus at . Transverse coordinate	0
target_y	m	position of target to focus at. Vertical coordinate	0
target_z	m	position of target to focus at. Straight ahead.	0
target_index	1	relative index of component to focus at, e.g. next is +1	0
F2	1	magnetic form factor squared	1
focus_r	m	Radius of sphere containing target.	0
focus_xw	m	horiz. dimension of a rectangular area	0
focus_yh	m	vert. dimension of a rectangular area	0
focus_aw	deg	horiz. angular dimension of a rectangular area	0
focus_ah	deg	vert. angular dimension of a rectangular area	0
verbose	1	flag for test printing (print if verbose==1)	0

Links

- Component source code found in file `Magnon_bcc.comp`.
- The test/example instrument Test_Magnon.instr.

9.10. The `Single_magnetic_crystal` McStas Component

Release: McStas 2.6

Mosaic magnetic single crystal with multiple scattering vectors.

Identification

- **Author:** Erik B Knudsen and Linda Udby
- **Origin:** DTU Physics
- **Date:** Jan 2020

Description

WARNING: This is an experimental component - no experimental validation has yet been done

Single magnetic crystal with mosaic. Delta-D/D option for finite-size effects. Multiple scattering and secondary extinction included. The mosaic may EITHER be specified isotropic by setting the mosaic input parameter, OR anisotropic by setting the mosaic_h, mosaic_v, and mosaic_n parameters.

The scattering is computed solely in an spin up-down configuration. That is the scattering is considered in relation to the externally defined vector (mx,my,mz), where it can be either SF or NSF. Simplifications and comments : Lande splitting factor is assumed to be g=2 Magnetic form factors are set =1

Sample shape: Sample shape may be a cylinder, a sphere, a box or any other shape

```
box/plate:       xwidth x yheight x zdepth
cylinder:        radius x yheight
sphere:          radius (yheight=0)
any shape:       geometry=OFF, PLY or STL file
```

The complex geometry option handles any closed non-convex polyhedra. It computes the intersection points of the neutron ray with the object transparently, so that it can be used like a regular sample object. It supports the OFF (incl. NOFF, COFF), PLY and STL file formats. A mesh must be a single, closed object: the neutron path through it is taken from the first entry and exit only, so a second part of the mesh further along the path (disjoint parts, or a concave shape the path re-enters) is not seen. Such files may be generated from XYZ data using qhull/powercrust, and viewed with geomview The default size of the object depends of the OFF file data, but its bounding box may be resized using xwidth,yheight and zdepth.

Also, always use a non-zero value of delta_d_d.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
atom_sites	str	File name containing the atoms present in the unit cell. Use empty ("") or NULL for sigma_inc scattering only	0
geometry	str	Name of an OFF, PLY or STL file for complex geometry. An OFF file may be generated from XYZ coordinates using qhull/powercrust	NULL
xwidth	m	Width of crystal	0
yheight	m	Height of crystal	0
zdepth	m	Depth of crystal (no extinction simulated)	0
radius	m	Outer radius of sample in (x,z) plane	0
delta_d_d	1	Lattice spacing variance, gaussian RMS	1e-4
mosaic	arcmin	Crystal mosaic (isotropic), gaussian RMS	-1
mosaic_h	arcmin	Horizontal (rotation around Y) mosaic (anisotropic), gaussian RMS	-1
mosaic_v	arcmin	Vertical (rotation around Z) mosaic (anisotropic), gaussian RMS	-1
mosaic_n	arcmin	Out-of-plane (Rotation around X) mosaic (anisotropic), gaussian RMS	-1
recip_cell	1	Choice of direct/reciprocal (0/1) unit cell definition	0
q_min	$Å^{-1}$	lower boundary of momentum transfer range to generate hkls in	0
q_max	$Å^{-1}$	upper boundary of momentum transfer range to generate hkls in	-1
mx	1	x-component of vector defining the SF/NSF direction	0
my	1	Coordinates of vector defining the SF/NSF direction	1
mz	1	z-component of vector defining the SF/NSF direction	0
na	1	Multiplier for unit cell vector a (see nb)	1
nb	1	Unit cell multipliers. The specified unit cell vectors are scaled by these factors. Note that the mulitpliers are applied directly to the raw input data. I.e. if recip. cell vectors are given, multipliers should be <1 (= 1/n). F.i. used to specify a magnetic unit cell which is larger than the chemical unit cell.	1

Name	Unit	Description	Default
nc	1	Multiplier for unit cell vector c (see nb)	1
ax	Å or $Å^{-1}$	x-coordinate of first (direct/recip) unit cell vector	0
ay	Å or $Å^{-1}$	Coordinates of first (direct/recip) unit cell vector	0
az	Å or $Å^{-1}$	z-coordinate of first (direct/recip) unit cell vector	0
bx	Å or $Å^{-1}$	x-coordinate of second (direct/recip) unit cell vector	0
by	Å or $Å^{-1}$	Coordinates of second (direct/recip) unit cell vector	0
bz	Å or $Å^{-1}$	z-coordinate of second (direct/recip) unit cell vector	0
cx	Å or $Å^{-1}$	x-coordinate of third (direct/recip) unit cell vector	0
cy	Å or $Å^{-1}$	Coordinates of third (direct/recip) unit cell vector	0
cz	Å or $Å^{-1}$	z-coordinate of third (direct/recip) unit cell vector	0
p_transmit	1	Monte Carlo probability for neutrons to be transmitted without any scattering. Used to improve statistics from weak reflections	-1
sigma_abs	barns	absorption cross-section per unit cell at 2200 m/s	0
sigma_inc	barns	incoherent scattering cross-section per unit cell	0
order	1	limit multiple scattering up to given order (0: all, 1: first, 2: second, ...)	0

Links

- Component source code found in file `Single_magnetic_crystal.comp`.

9.11. The `NCrystal_sample` McStas Component

McStas sample component for the NCrystal library for thermal neutron transport (www).

Identification

- **Author:** NCrystal developers

- **Origin:** NCrystal Developers (European Spallation Source ERIC and DTU Nutech)
- **Date:** 2015

Description

McStas sample component using the NCrystal library for thermal neutron transport in a single bulk material filling a single simple convex volume (box, sphere or cylinder).

The geometrical layout of the volume is determined via the xwidth, yheight, zdepth, and radius parameters, and the material being modelled is determined via an NCrystal configuration string ("cfg-string").

For more information about NCrystal and cfg-strings, refer to the NCrystal wiki. In particular, browse the available datafiles at Data-library and read about the format of the cfg-string expected in the "cfg" parameter at Using-NCrystal.

For more complicated geometries, it might be desirable to use NCrystal via the McStas Union components instead.

Note that the physics backend of this component will be whichever NCrystal installation is available and associated with the "ncrystal-config" command.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
cfg	str	NCrystal material configuration string (details on this page).	"void"
xwidth	m	x-dimension (width) of sample, if box shape is desired	0
yheight	m	y-dimension (height) of sample, if box or cylinder shape is desired	0
zdepth	m	z-dimension (depth) of sample, if box shape is desired	0
radius	m	radius of sample, if sphere or cylinder shape is desired	0
absorptionmode	0|1|2	0 : disable absorption. 1 : enable absorption via intensity reduction. 2 : enable absorption by terminating the tracking.	1
multscat	0|1	0 : disable multiple scattering. 1 : enable multiple scattering	1

Links

- Component source code found in file `NCrystal_sample.comp`.

- The NCrystal wiki at https://github.com/mctools/ncrystal/wiki.

9.12. Samples for resolution-function calculations

9.13. The `Res_sample` McStas Component

Sample for resolution function calculation.

Identification

- **Author:** Kristian Nielsen
- **Origin:** Risoe
- **Date:** 1999

Description

An inelastic sample with completely uniform scattering in both Q and energy. This sample is used together with the Res_monitor component and (optionally) the mcresplot front-end to compute the resolution function of triple-axis or inverse-geometry time-of-flight instruments.

The shape of the sample is either a hollow cylinder or a rectangular box. The hollow cylinder shape is specified with an inner and outer radius. It has no lids: the cavity spans the full yheight, as a sample can without lids. The box is specified with dimensions xwidth, yheight, zdepth.

The scattered neutrons will have directions towards a given disk and energies betweed E0-dE and E0+dE. This target area may also be rectangular if specified focus_xw and focus_yh or focus_aw and focus_ah, respectively in meters and degrees. The target itself is either situated according to given coordinates (x,y,z), or setting the relative target_index of the component to focus at (next is +1). This target position will be set to its AT position. When targeting to centered components, such as spheres or cylinders, define an Arm component where to focus at.

Example: Res_sample(thickness=0.001,radius=0.02,yheight=0.4,focus_r=0.05, E0=14.6,dE=2, target_x=0, target_y=0, target_z=1)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
thickness	m	Thickness of hollow cylinder in (x,z) plane	0
radius	m	Outer radius of hollow cylinder	0.01
focus_r	m	Radius of sphere containing target.	0.05
E0	meV	Center of scattered energy range	14
dE	meV	half width of scattered energy range	2

Name	Unit	Description	Default
target_x	m	X-position of target to focus at [m]	0
target_y	m	Y-position of target to focus at	0
target_z	m	Z-position of target to focus at [m]	.5
focus_xw	m	horiz. dimension of a rectangular area	0
focus_yh	m	vert. dimension of a rectangular area	0
focus_aw	deg	horiz. angular dimension of a rectangular area	0
focus_ah	deg	vert. angular dimension of a rectangular area	0
xwidth	m	horiz. dimension of sample, as a width	0
yheight	m	vert. dimension of sample, as a height	0.05
zdepth	m	depth of sample	0
target_index	1	relative index of component to focus at, e.g. next is +1	0

Links

- Component source code found in file `Res_sample.comp`.

9.14. The `TOFRes_sample` McStas Component

Sample for TOF resolution function calculation.

Identification

- **Author:** KL, 10 October 2004
- **Origin:** Risoe
- **Date:** 1999

Description

An inelastic sample with completely uniform scattering in both solid angle and energy. This sample is used together with the TOFRes_monitor component and (optionally) the mcresplot front-end to compute the resolution function of all time-of-flight instruments. The method of time focusing is used to optimize the simulations.

The shape of the sample is either: 1. Hollow cylinder (Please note that the cylinder **must** be specified with both a radius and a wall-thickness!). It has no lids: the cavity spans the full yheight, as a sample can without lids. 2. A massive, rectangular box specified with dimensions xwidth, yheight, zdepth. (i.e. **withouht** a thickness)

hollow cylinder shape **must** be specified with both a radius and a wall-thickness. The box is

The scattered neutrons will have directions towards a given target and detector arrival time in an interval of time_width centered on time_bin. This target area is default disk shaped, but may also be rectangular if specified focus_xw and focus_yh or focus_aw and focus_ah, respectively in meters and degrees. The target itself is either situated according to given coordinates (x,y,z), or setting the relative target_index of the component to focus at (next is +1). This target position will be set to its AT position. When targeting to centered components, such as spheres or cylinders, define an Arm component where to focus at.

Example: TOFRes_sample(thickness=0.001, radius=0.01, yheight=0.04, focus_xw=0.025, focus_yh=0.025, time_bin=3e4, time_width=200, target_x=0, target_y=0, target_z=1)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
thickness	m	Thickness of hollow cylinder in (x,z) plane	0
radius	m	Outer radius of hollow cylinder	0.01
yheight	m	vert. dimension of sample, as a height	0.05
focus_r	m	Radius of sphere containing target	0.05
time_bin	us	position of time bin	20000
time_width	us	width of time bin	10
f	1	Adaptive time-shortening factor	50
target_x	m	position of target to focus at	0
target_y	m	position of target to focus at	0
target_z	m	position of target to focus at	.5
focus_xw	m	horiz. dimension of a rectangular area	0
focus_yh	m	vert. dimension of a rectangular area	0
focus_aw	deg	horiz. angular dimension of a rectangular area	0
focus_ah	deg	vert. angular dimension of a rectangular area	0
xwidth	m	horiz. dimension of sample, as a width	0
zdepth	m	depth of sample	0
target_index	1	relative index of component to focus at, e.g. next is +1	0

Links

- Component source code found in file `TOFRes_sample.comp`.

10. Monitors and detectors

In real neutron experiments, detectors and monitors play quite different roles. One wants the detectors to be as efficient as possible, counting all neutrons (absorbing them in the process), while the monitors measure the intensity of the incoming beam, and must as such be almost transparent, interacting only with (roughly) 0.1-1% of the neutrons passing by. In computer simulations, it is of course possible to detect every neutron ray without absorbing it or disturbing any of its parameters. Hence, the two components have very similar functions in the simulations, and we do not distinguish between them. For simplicity, they are from here on just called **monitors**.

Another important difference between computer simulations and real experiments is that one may allow the monitor to be sensitive to any neutron property, as *e.g.* direction, energy, and divergence, in addition to what is found in real-world detectors (space and time). One may, in fact, let the monitor record correlations between these properties, as seen for example in the divergence/position sensitive monitor in section 10.7.

When a monitor detects a neutron ray, a number counting variable is incremented: $n_i = n_{i-1}+1$. In addition, the neutron weight p_i is added to the weight counting variable: $I_i = I_{i-1} + p_i$, and the second moment of the weight is updated: $M_{2,i} = M_{2,i-1} + p_i^2$. As also discussed chapter 2, after a simulation of N rays the detected intensity (in units of neutrons/sec.) is I_N, while the estimated errorbar is $\sqrt{M_{2,N}^2}$.

Many different monitor components have been developed for McStas, but we have decided to support only the most important ones. One example of the monitors we have omitted is the single monitor, **Monitor**, that measures just one number (with errorbars) per simulation. This effect is mirrored by any of the 1- or 2-dimensional components we support, e.g. the PSD_monitor. In case additional functionality of monitors is required, the few code lines of existing monitors can easily be modified.

However, the ultimate solution is the use of the "Swiss army knife" of monitors, **Monitor_nD**, that can face almost any simulation requirement, but will prove challenging for users who like to perform own modifications. When compiling an instrument for GPU/OpenACC (see the User Manual, section on GPU acceleration), `Monitor_nD` automatically falls back to the companion component **Monitor_nD_noacc** for any `options` string not yet supported on GPU, via the `NOACC`/`CPU` funneling mechanism (see the User Manual's kernel chapter); this is handled transparently and does not normally require any instrument change.

The monitors presented in detail below are a representative selection; the library additionally contains cylindrical and spherical variants of most of them (e.g. `Cyl_monitor`, `PSD_monitor_4PI`), energy-transfer and $S(q,\omega)$-aware monitors (`Sqw_monitor`, `TOFLambda_monitor`), polarisation-aware monitors (`Pol_monitor`, `PolLambda_monitor`), and flexible, user-configurable

histogrammers (`Flex_monitor_1D/2D/3D`). Use `mcdoc` (User Manual, section on McDoc) for the complete, current list.

10.1. The `TOF_monitor` McStas Component

Rectangular Time-of-flight monitor.

Identification

- **Author:** KN, M. Hagen
- **Origin:** Risoe
- **Date:** August 1998

Description

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nt	1	Number of time bins	20
filename	string	Name of file in which to store the detector image	0
xmin	m	Lower x bound of detector opening	-0.05
xmax	m	Upper x bound of detector opening	0.05
ymin	m	Lower y bound of detector opening	-0.05
ymax	m	Upper y bound of detector opening	0.05
xwidth	m	Width of detector. Overrides xmin, xmax	0
yheight	m	Height of detector. Overrides ymin, ymax	0
tmin	mu-s	Lower time limit	0
tmax	mu-s	Upper time limit	0
dt	mu-s	Length of each time bin	1.0
restore_neutron	1	If set, the monitor does not influence the neutron state	0
nowritefile	1	If set, monitor will skip writing to disk	0

Links

- Component source code found in file `TOF_monitor.comp`.

The time-of-flight monitor

The component **TOF_monitor** has a rectangular opening in the (x, y) plane, given by the x and y parameters, like for **Slit**. The neutron ray is propagated to the plane of the

monitor by the kernel call PROP_Z0. A neutron ray is counted if it passes within the rectangular opening given by the x and y limits.

Special about **TOF_monitor** is that it is sensitive to the arrival time, t, of the neutron ray. Like in a real time-of-flight detector, the time dimension is binned into small time intervals. Hence this monitor maintains a one-dimensional histogram of counts. The n_{chan} time intervals begin at t_0 and end at t_1 (alternatively, the interval length is specified by Δt). As usual in time-of-flight analysis, all times are given in units of μs.

The output parameters from **TOF_monitor** are the three count numbers, N, I, and M_2 for the total counts in the monitor. In addition, a file, `filename`, is produced with a list of the same three data divided in different TOF bins. This file can be read and plotted by the mcplot tool; see the System Manual.

10.2. The `TOF2E_monitor` McStas Component

TOF-sensitive monitor, converting to energy

Identification

- **Author:** Kim Lefmann and Helmuth Schoeber
- **Origin:** Risoe
- **Date:** Sept. 13, 2006

Description

A square single monitor that measures the energy of the incoming neutrons from their time-of-flight

Example: TOF2E_monitor(xmin=-0.1, xmax=0.1, ymin=-0.1, ymax=0.1, Emin=1, Emax=50, nE=20, filename="Output.nrj", L_flight=20, T_zero=0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nE	1	Number of energy channels	20
filename	string	Name of file in which to store the detector image	0
nowritefile	1	If set, monitor will skip writing to disk	0
xmin	m	Lower x bound of detector opening	-0.05
xmax	m	Upper x bound of detector opening	0.05
ymin	m	Lower y bound of detector opening	-0.05
ymax	m	Upper y bound of detector opening	0.05

Name	Unit	Description	Default
xwidth	m	Width of detector. Overrides xmin, xmax	0
yheight	m	Height of detector. Overrides ymin, ymax	0
Emin	meV	Minimum energy to detect	
Emax	meV	Maximum energy to detect	
T_zero	s	Zero point in time	
L_flight	m	flight length user in conversion	
restore_neutron	1	If set, the monitor does not influence the neutron state	0

Links

- Component source code found in file `TOF2E_monitor.comp`.

A time-of-flight monitor with simple energy analysis

The component **TOF2E_monitor** resembles **TOF_monitor** to a very large extent. Only this monitor converts the neutron flight time to energy - as would be done in an experiment. The *apparent* neutron energy, E_{app} is calculated from the apparent velocity, given by

$$v_{\text{app}} = \frac{L_{\text{flight}}}{t - t_0}, \tag{10.1}$$

where the time offset, t_0 defaults to zero. E_{app} is binned in *nchan* bins between E_{min} and E_{max} (in meV).

The output parameters from **TOF2E_monitor** are the total counts, and a file with 1-dimensional data vs. E_{app}, similar to **TOF_monitor**.

10.3. The `E_monitor` McStas Component

Energy-sensitive monitor.

Identification

- **Author:** Kristian Nielsen and Kim Lefmann
- **Origin:** Risoe
- **Date:** April 20, 1998

Description

A square single monitor that measures the energy of the incoming neutrons.

Example: E_monitor(xmin=-0.1, xmax=0.1, ymin=-0.1, ymax=0.1, Emin=1, Emax=50, nE=20, filename="Output.nrj")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nE	1	Number of energy channels	20
filename	string	Name of file in which to store the detector image	0
xmin	m	Lower x bound of detector opening	-0.05
xmax	m	Upper x bound of detector opening	0.05
ymin	m	Lower y bound of detector opening	-0.05
ymax	m	Upper y bound of detector opening	0.05
nowritefile	1	If set, monitor will skip writing to disk	0
xwidth	m	Width of detector. Overrides xmin, xmax.	0
yheight	m	Height of detector. Overrides ymin, ymax.	0
Emin	meV	Minimum energy to detect	
Emax	meV	Maximum energy to detect	
restore_neutron	1	If set, the monitor does not influence the neutron state	0

Links

- Component source code found in file `E_monitor.comp`.

The energy-sensitive monitor

The component **E_monitor** resembles **TOF_monitor** to a very large extent. Only this monitor is sensitive to the neutron energy, which in binned in nE bins between $E_{\min}$ and $E_{\max}$ (in meV).

The output parameters from **E_monitor** are the total counts, and a file with 1-dimensional data vs. E, similar to **TOF_monitor**.

10.4. The `L_monitor` McStas Component

Wavelength-sensitive monitor.

Identification

- **Author:** Kristian Nielsen and Kim Lefmann
- **Origin:** Risoe
- **Date:** April 20, 1998

Description

A square single monitor that measures the wavelength of the incoming neutrons.
Example: L_monitor(xmin=-0.1, xmax=0.1, ymin=-0.1, ymax=0.1, nL=20, filename="Output.L", Lmin=2, Lmax=10)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nL	1	Number of wavelength channels	20
filename	string	Name of file in which to store the detector image	0
nowritefile	1	If set, monitor will skip writing to disk	0
xmin	m	Lower x bound of detector opening	-0.05
xmax	m	Upper x bound of detector opening	0.05
ymin	m	Lower y bound of detector opening	-0.05
ymax	m	Upper y bound of detector opening	0.05
xwidth	m	Width of detector. Overrides xmin, xmax.	0
yheight	m	Height of detector. Overrides ymin, ymax.	0
Lmin	Å	Minimum wavelength to detect	
Lmax	Å	Maximum wavelength to detect	
restore_neutron	1	If set, the monitor does not influence the neutron state	0

Links

- Component source code found in file `L_monitor.comp`.

The wavelength sensitive monitor

The component **L_monitor** is very similar to **TOF_monitor** and **E_monitor**. This component is just sensitive to the neutron wavelength. The wavelength spectrum is output in a one-dimensional histogram. between $\lambda_{\min}$ and $\lambda_{\max}$ (measured in Å).

As for the two other 1-dimensional monitors, this component outputs the total counts and a file with the histogram.

10.5. The `PSD_monitor` McStas Component

Position-sensitive monitor.

Identification

- **Author:** Kim Lefmann
- **Origin:** Risoe
- **Date:** Feb 3, 1998

Description

An (n times m) pixel PSD monitor. This component may also be used as a beam detector.
Example: PSD_monitor(xmin=-0.1, xmax=0.1, ymin=-0.1, ymax=0.1, nx=90, ny=90, filename="Output.psd")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nx	1	Number of pixel columns	90
ny	1	Number of pixel rows	90
filename	string	Name of file in which to store the detector image	0
xmin	m	Lower x bound of detector opening	-0.05
xmax	m	Upper x bound of detector opening	0.05
ymin	m	Lower y bound of detector opening	-0.05
ymax	m	Upper y bound of detector opening	0.05
xwidth	m	Width of detector. Overrides xmin, xmax	0
yheight	m	Height of detector. Overrides ymin, ymax	0
restore_neutron	1	If set, the monitor does not influence the neutron state	0
nowritefile	1	If set, monitor will skip writing to disk	0

Links

- Component source code found in file `PSD_monitor.comp`.

The PSD monitor

The component **PSD_monitor** resembles other monitors, e.g. **TOF_Monitor**, and also propagates the neutron ray to the detector surface in the (x, y)-plane, where the detector window is set by the x and y input coordinates. The PSD monitor, though, is not sensitive to the arrival time of the neutron ray, but rather to its position. The rectangular monitor window, given by the x and y limits is divided into $n_x \times n_y$ pixels.

The output from **PSD_monitor** is the integrated counts, n, I, M_2, as well as three two-dimensional arrays of counts: $n(x, y), I(x, y), M_2(x, y)$. The arrays are written to a file, `filename`, and can be read e.g. by the tool **mcplot**, see the system manual.

10.6. The `Divergence_monitor` McStas Component

Horizontal+vertical divergence monitor.

Identification

- **Author:** Kim Lefmann
- **Origin:** Risoe
- **Date:** Nov. 11, 1998

Description

A divergence sensitive monitor. The counts are distributed in (n times m) pixels.
Example: Divergence_monitor(nh=20, nv=20, filename="Output.pos",

```
xmin=-0.1, xmax=0.1, ymin=-0.1, ymax=0.1,
```

maxdiv_h=2, maxdiv_v=2)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nh	1	Number of pixel rows	20
nv	1	Number of pixel columns	20
filename	str	Name of file in which to store the detector image text	0
xmin	m	Lower x bound of detector opening	-0.05
xmax	m	Upper x bound of detector opening	0.05
ymin	m	Lower y bound of detector opening	-0.05
ymax	m	Upper y bound of detector opening	0.05

Name	Unit	Description	Default
nowritefile	1	If set, monitor will skip writing to disk	0
xwidth	m	Width of detector. Overrides xmin, xmax	0
yheight	m	Height of detector. Overrides ymin, ymax	0
maxdiv_h	degrees	Maximal vertical divergence detected	2
maxdiv_v	degrees	Maximal vertical divergence detected	2
restore_neutron	1	If set, the monitor does not influence the neutron state	0
nx	1	x-component of the "forward" direction vector (see ny)	0
ny	1	Vector definition of "forward" direction wrt. divergence, to be used e.g. when the monitor is rotated into the horizontal plane	0
nz	1	z-component of the "forward" direction vector (see ny)	1

Links

- Component source code found in file `Divergence_monitor.comp`.

A divergence sensitive monitor

The component **Divergence_monitor** is a two-dimensional monitor, which resembles **PSD_Monitor**. As for this component, the detector window is set by the x and y input coordinates. **Divergence_ monitor** is sensitive to the neutron divergence, defined by $\eta_{\mathrm{h}} = \tan^{-1}(v_x/v_z)$ and $\eta_{\mathrm{v}} = \tan^{-1}(v_y/v_z)$. The neutron counts are being histogrammed into $n_{\mathrm{v}} \times n_{\mathrm{h}}$ pixels. The divergence range accepted is in the vertical direction $[-\eta_{\mathrm{v,max}}; \eta_{\mathrm{v,max}}]$, and similar for the horizontal direction.

The output from **PSD_monitor** is the integrated counts, n, I, M_2, as well as three two-dimensional arrays of counts: $n(\eta_{\mathrm{v}}, \eta_{\mathrm{h}}), I(\eta_{\mathrm{v}}, \eta_{\mathrm{h}}), M_2(\eta_{\mathrm{v}}, \eta_{\mathrm{h}})$. The arrays are written to a file, `filename`, and can be read e.g. by the tool **MC_plot**, see the system manual.

10.7. The `DivPos_monitor` McStas Component

Divergence/position monitor (acceptance diagram).

Identification

- **Author:** Kristian Nielsen
- **Origin:** Risoe

- **Date:** 1999

Description

2D detector for intensity as a function of position and divergence, either horizontally or vertically (depending on the flag vertical). This gives information similar to an aceptance diagram used eg. to investigate beam profiles in neutron guides.

Example: DivPos_monitor(nh=20, ndiv=20, filename="Output.dip", xmin=-0.1, xmax=0.1, ymin=-0.1, ymax=0.1, maxdiv_h=2)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nb	1	Number of bins in position.	20
ndiv	1	Number of bins in divergence.	20
filename	string	Name of file in which to store the detector image.	0
xmin	m	Lower x bound of detector opening	-0.05
xmax	m	Upper x bound of detector opening	0.05
ymin	m	Lower y bound of detector opening	-0.05
ymax	m	Upper y bound of detector opening	0.05
xwidth	m	Width of detector. Overrides xmin,xmax.	0
yheight	m	Height of detector. Overrides ymin,ymax.	0
maxdiv	degrees	Maximal divergence detected.	2
restore_neutron	1	If set, the monitor does not influence the neutron state.	0
nx	1	x-component of the "forward" direction vector (see ny)	0
ny	1	Vector definition of "forward" direction wrt. divergence, to be used e.g. when the monitor is rotated into the horizontal plane	0
nz	1	z-component of the "forward" direction vector (see ny)	1
vertical	1	Monitor intensity as a function of vertical divergence and position.	0
nowritefile	1	If set, monitor will skip writing to disk CALCULATED PARAMETERS:	0

Links

- Component source code found in file `DivPos_monitor.comp`.

A divergence and position sensitive monitor

DivPos_monitor is a two-dimensional monitor component, which is sensitive to both horizontal position (x) and horizontal divergence defined by $\eta_\mathrm{h} = \tan^{-1}(v_x/v_z)$. The detector window is set by the x and y input coordinates.

The neutron counts are being histogrammed into $n_x \times n_\mathrm{h}$ pixels. The horizontal divergence range accepted is $[-\eta_{\mathrm{h,max}}; \eta_{\mathrm{h,max}}]$, and the horizontal position range is the size of the detector.

The output from **PSD_monitor** is the integrated counts, n, I, M_2, as well as three two-dimensional arrays of counts: $n(x, \eta_\mathrm{h}), I(x, \eta_\mathrm{h}), M_2(x, \eta_\mathrm{h})$. The arrays are written to a file and can be read e.g. by the tool **mcplot**, see the system manual.

This component can be used for measuring acceptance diagrams [Cus03]. **PSD_monitor** can easily be changed into being sensitive to y and vertical divergence by a 90 degree rotation around the z-axis.

10.8. The `Monitor_nD` McStas Component

Release: McStas 1.6 Version: $Revision$

This component is a general Monitor that can output 0/1/2D signals (Intensity or signal vs. [something] and vs. [something] ...)

Identification

- **Author:** Emmanuel Farhi
- **Origin:** ILL
- **Date:** 14th Feb 2000.

Description

This component is a general Monitor that can output 0/1/2D signals It can produce many 1D signals (one for any variable specified in option list), or a single 2D output (two variables correlation). Also, an additional 'list' of neutron events can be produced. By default, monitor is square (in x/y plane). A disk shape is also possible The 'cylinder' and 'banana' option will change that for a banana shape The 'sphere' option simulates spherical detector. The 'box' is a box. The cylinder, sphere and banana should be centered on the scattering point. The 'geometry' parameter uses an OFF, PLY or STL mesh file as the detector surface instead, and overrides the shape options. Non-zero xwidth, yheight, zdepth resize the mesh bounding box (a single one keeps the proportions); the mesh is not re-centred. With the 'mantid' option, each face of the file is one pixel (pixel id = face index + pixel min). The monitored flux may be per monitor unit area, and weighted by a lambda/lambda(2200m/s) factor to obtain standard integrated capture flux. In normal configuration, the Monitor_nD measures the current parameters of the neutron that is being detected. USERVARS may be used in order to study correlations between a neutron being detected in a Monitor_nD place, and given parameters that are monitored elsewhere (at the point of initialisation of the USERVARS). The monitor can also act as a 3He gas detector, taking into account the detection efficiency.

The 'bins' and 'limits' modifiers are to be used after each variable, and 'auto','log' and 'abs' come before it. (eg: auto abs log hdiv bins=10 limits=[-5 5]) When placed after all variables, these two latter modifiers apply to the signal (e.g. intensity). Unknown keywords are ignored. If no limits are specified for a given observable, reasonable defaults will be applied. Note that these implicit limits are **even** applied in list mode.

Implicit limits for typical variables: (consult monitor_nd-lib.c if you don't find your variable here) x, y, z: Derived from detection-object geometry

```
k: [0 10] Angs-1
v: [0 1e6] m/s
t: [0 1] s
```

p: [0 FLT_MAX] in intensity-units

```
vx, vy: [-1000 1000] m/s
vz: [0 10000] m/s
kx, ky: [-1 1] Angs-1
kz: [-10 10] Angs-1
```

energy, omega: [0 100] meV lambda,wavelength: [0 100] Å sx, sy, sz: [-1 1] in polarisation-units angle: [-50 50] deg divergence, vdiv, hdiv, xdiv, ydiv: [-5 5] deg longitude, lattitude: [-180 180] deg neutron: [0 simulaton_ncount] id, pixel id: [0 FLT_MAX]

```
uservars u0,u1,u2,u3,u4,u5,u6,u7,u8,u9: [-1e10 1e10]
```

In the case of multiple components at the same position, the 'parallel' keyword must be used in each instance instead of defining a GROUP.

Possible options are Variables to record: kx ky kz k wavevector [Å-1] Wavevector on x,y,z and norm

```
vx vy vz v            [m/s]    Velocity on x,y,z and norm
x y z radius          [m]      Distance, Position and norm
xy, yz, xz            [m]      Radial position in xy, yz and xz plane
kxy kyz kxz           [Angs-1] Radial wavevector in xy, yz and xz plane
vxy vyz vxz           [m/s]    Radial velocity in xy, yz and xz plane
t time                [s]      Time of Flight
energy omega          [meV]    energy of neutron
lambda wavelength     [Angs]   wavelength of neutron
sx sy sz              [1]      Spin
vdiv ydiv dy          [deg]    vertical divergence (y)
hdiv divergence xdiv  [deg]    horizontal divergence (x)
angle                 [deg]    divergence from <z> direction
theta longitude       [deg]    longitude (x/z) for sphere and cylinder
phi   lattitude       [deg]    lattitude (y/z) for sphere and cylinder

user0 user1           will monitor the [Mon_Name]_Vars.UserVariable{0|1|2|3|4|5}
user2 user3           to be assigned in an other component (see below)
```

user4 user5 user6 user7 user8 user9

```
Premonitoring:            Please use uservars in place of the former PreMonitor_nD.

p i                   [n/s  or  n/cm^2/s]  ('intensity', 'flux': name the default sig
ncounts n neutron     [1]      neutron ID, i.e current event index
pixel id              [1]      pixelID in histogram made of preceeding vars, e.g. 'th
```

Other options keywords are:

```
abs                         Will monitor the abs of the following variable or of the signal (if u
auto                        Automatically set detector limits for one/all
all  {limits|bins|auto}     To set all limits or bins values or auto mode
binary {float|double}       with 'source' option, saves in compact files
bins=[bins=20]              Number of bins in the detector along dimension
borders                     To also count off-limits neutrons (X < min or X > max)
capture                     weight by lambda/lambda(2200m/s) capture flux
file=string                 Detector image file name. default is component name, plus date and va
incoming                    Monitor incoming beam in non flat det
limits=[min max]            Lower/Upper limits for axes (see up for the variable unit)
```

list=[counts=1000] or all For a long file of neutron characteristics with [counts] or all events

```
log                         Will monitor the log of the following variable or of the signal (if u
min=[min_value]             Same as limits, but only sets the min or max
```

max=[max_value]

```
multiple                    Create multiple independant 1D monitors files
no or not                   Revert next option
outgoing                    Monitor outgoing beam (default)
parallel                    Use this option when the next component is at the same position (para
per cm2                     Intensity will be per cm^2 (detector area). Displays beam section.
per steradian               Displays beam solid angle in steradian
signal=[var]                Will monitor [var] instead of usual intensity
slit or absorb              Absorb neutrons that are out detector
source                      The monitor will save neutron states
inactivate                  To inactivate detector (0D detector)
verbose                     To display additional informations
```

3He_pressure=[3 in bars] The 3He gas pressure in detector. 3He_pressure=0 is perfect detector (default)

Detector shape options (specified as xwidth,yheight,zdepth or x/y/z/min/max)

```
box                         Box of size xwidth, yheight, zdepth.
cylinder                    To get a cylindrical monitor (diameter is xwidth or set radius, heigh
banana                      Same as cylinder, without top/bottom, on restricted angular area; use
disk                        Disk flat xy monitor. diameter is xwidth.
sphere                      To get a spherical monitor (e.g. a 4PI) (diameter is xwidth or set ra
square                      Square flat xy monitor (xwidth, yheight).
previous                    The monitor uses PREVIOUS component as detector surface.
geometry="file"             Mesh file (OFF, PLY or STL) as detector surface. A parameter, not an
```

EXAMPLES: MyMon = Monitor_nD(xwidth = 0.1, yheight = 0.1, zdepth = 0,   options = "intensity per cm2 angle,limits=[-5 5] bins=10,with   borde file = mon1"); will monitor neutron angle from [z] axis, between -5 and 5 degrees, in 10 bins, into "mon1.A" output 1D file

 options = "sphere theta phi outgoing" for a sphere PSD detector (out beam) and saves into file "MyMon_[Date_ID].th_ph"

 options = "banana, theta limits=[10,130], bins=120, y" a theta/height banana detector

 options = "angle radius all auto" is a 2D monitor with automatic limits

 options = "list=1000 kx ky kz energy" records 1000 neutron event in a file

 options = "multiple kx ky kz, auto abs log t, and list all neutrons" makes 4 output 1D files and produces a complete list for all neutrons and monitor log(abs(tof)) within automatic limits (for t)

 options = "theta y, sphere, pixel min=100" a 4pi detector which outputs an event list with pixelID from the actual detector surface, starting from index 100.

 To dynamically define a number of bins, or limits:

```
Use in DECLARE:    char op[256];
```

Use in INITIALIZE: sprintf(op, "lambda limits=[%g %g], bins=%i", lmin, lmax, lbin);

```
Use in TRACE:      Monitor_nD(... options=op ...)
```

How to monitor any instrument/component variable into a Monitor_nD Suppose you want to monitor a variable 'age' which you assign somwhere in the instrument: COMPONENT MyMonitor = Monitor_nD(xwidth = 0.1, yheight = 0.1, user1="age", username1="Age of the Captain [years]", options="user1, auto")

```
AT ...
```

%BUGS The 'auto' option for guessing optimal variable bounds should NOT be used with MPI as each process may use different limits.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
user0	str	Variable name of USERVAR to be monitored by user0.	""
user1	str	Variable name of USERVAR to be monitored by user1.	""
user2	str	Variable name of USERVAR to be monitored by user2.	""

Name	Unit	Description	Default
user3	str	Variable name of USERVAR to be monitored by user3.	""
user4	str	Variable name of USERVAR to be monitored by user4.	""
user5	str	Variable name of USERVAR to be monitored by user5.	""
user6	str	Variable name of USERVAR to be monitored by user6.	""
user7	str	Variable name of USERVAR to be monitored by user7.	""
user8	str	Variable name of USERVAR to be monitored by user8.	""
user9	str	Variable name of USERVAR to be monitored by user9.	""
xwidth	m	Width of detector.	0
yheight	m	Height of detector.	0
zdepth	m	Thickness of detector (z).	0
xmin	m	Lower x bound of opening	0
xmax	m	Upper x bound of opening	0
ymin	m	Lower y bound of opening	0
ymax	m	Upper y bound of opening	0
zmin	m	Lower z bound of opening	0
zmax	m	Upper z bound of opening	0
bins	1	Number of bins to force for all variables. Use 'bins' keyword in 'options' for heterogeneous bins	0
min	u	Minimum range value to force for all variables. Use 'min' or 'limits' keyword in 'options' for other limits	-1e40
max	u	Maximum range value to force for all variables. Use 'max' or 'limits' keyword in 'options' for other limits	1e40
restore_neutron	0|1	If set, the monitor does not influence the neutron state. Equivalent to setting the 'parallel' option.	0
radius	m	Radius of sphere/banana shape monitor	0
options	str	String that specifies the configuration of the monitor. The general syntax is "[x] options..." (see Descr.).	"NULL"
filename	str	Output file name (overrides file=XX option).	"NULL"

Name	Unit	Description	Default
geometry	str	Name of an OFF, PLY or STL mesh file to use as detector surface (overrides the shape options)	"NULL"
nowritefile	1	If set, monitor will skip writing to disk	0
nexus_bins	1	NeXus mode only: store component BIN information (-1 disable, 0 enable for list mode monitor, 1 enable for any montor)	0
username0	str	Name assigned to User0	"NULL"
username1	str	Name assigned to User1	"NULL"
username2	str	Name assigned to User2	"NULL"
username3	str	Name assigned to User3	"NULL"
username4	str	Name assigned to User4	"NULL"
username5	str	Name assigned to User5	"NULL"
username6	str	Name assigned to User6	"NULL"
username7	str	Name assigned to User7	"NULL"
username8	str	Name assigned to User8	"NULL"
username9	str	Name assigned to User9	"NULL"

Links

- Component source code found in file `Monitor_nD.comp`.

A general Monitor for 0D/1D/2D records

The component **Monitor_nD** is a general Monitor that may output any set of physical parameters regarding the passing neutrons. The generated files are either a set of 1D signals ([Intensity] *vs.* [Variable]), or a single 2D signal ([Intensity] *vs.* [Variable 1] *vs.* [Variable 1]), and possibly a simple long list of selected physical parameters for each neutron.

The input parameters for **Monitor_nD** are its dimensions $x_{\min}, x_{\max}, y_{\min}, y_{\max}$ (in meters) and an *options* string describing what to detect, and what to do with the signals, in clear language. The $x_{width}, y_{height}, z_{depth}$ may also be used to enter dimensions.

Eventhough the possibilities of Monitor_nD are numerous, its usage remains as simple as possible, specially in the `options` parameter, which 'understands' normal language. The formatting of the *options* parameter is free, as long as it contains some specific keywords, that can be sometimes followed by values. The *no* or *not* option modifier will revert next option. The *all* option can also affect a set of monitor configuration parameters (see below).

As the usage of this component enables to monitor virtually anything, and thus the combinations of options and parameters is infinite, we shall only present the most basic configuration. The reader should refer to the on-line component help, using e.g.

`mcdoc Monitor_nD.comp`.

The Monitor_nD geometry

The monitor shape can be selected among seven geometries:

1. (*square*) The default geometry is flat rectangular in (xy) plane with dimensions $x_{\min}, x_{\max}, y_{\min}$, $y_{\max}$, or x_{width}, y_{height}.

2. (*box*) A rectangular box with dimensions $x_{width}, y_{height}, z_{depth}$.

3. (*disk*) When choosing this geometry, the detector is a flat disk in (xy) plane. The radius is then

$$\text{radius} = \max(\text{abs}\ [x_{\min}, x_{\max}, y_{\ \min}, y_{\max}, x_{width}/2, y_{height}/2]). \tag{10.2}$$

4. (*sphere*) The detector is a sphere with the same radius as for the *disk* geometry.

5. (*cylinder*) The detector is a cylinder with revolution axis along y (vertical). The radius in (xz) plane is

$$\text{radius} = \max(\text{abs}\ [x_{\min}, x_{\max}, x_{width}/2]), \tag{10.3}$$

   and the height along y is

$$\text{height} = |y_{\max} - y_{\max}| \text{or} y_{height}. \tag{10.4}$$

6. (*banana*) The same as the cylinder, but without the top/bottom caps, and on a restricted angular range. The angular range is specified using a `theta` variable limit specification in the `options`.

7. (*previous*) The detector has the shape of the previous component. This may be a surface or a volume. In this case, the neutron is detected on previous component, and there is not neutron propagation.

By default, the monitor is flat, rectangular. Of course, you can choose the orientation of the **Monitor_nD** in the instrument description file with the usual `ROTATED` modifier.

For the *box*, *sphere* and *cylinder*, the outgoing neutrons are monitored by default, but you can choose to monitor incoming neutron with the *incoming* option.

At last, the *slit* or *absorb* option will ask the component to absorb the neutrons that do not intersect the monitor. The *exclusive* option word removes neutrons which are similarly outside the monitor limits (that may be other than geometrical).

The *parallel* option keyword is of common use in the case where the **Monitor_nD** is superposed with other components. It ensures that neutrons are detected independently of other geometrical constrains. This is generally the case when you need e.g. to place more than one monitor at the same place.

The neutron parameters that can be monitored

There are many different variables that can be monitored at the same time and position. Some can have more than one name (e.g. `energy` or `omega`).

```
kx ky kz k wavevector [Angs-1] (     usually axis are
vx vy vz v            [m/s]          x=horz., y=vert., z=on axis)
x y z                 [m]       Distance, Position
kxy vxy xy radius     [m]       Radial wavevector, velocity and position
t time                [s]       Time of Flight
energy omega          [meV]
lambda wavelength     [Angs]
p intensity flux      [n/s] or [n/cm^2/s]
ncounts               [1]
sx sy sz              [1]       Spin
vdiv ydiv dy          [deg]     vertical divergence (y)
hdiv divergence xdiv  [deg]     horizontal divergence (x)
angle                 [deg]     divergence from  direction
theta longitude       [deg]     longitude (x/z) [for sphere and cylinder]
phi   lattitude       [deg]     lattitude (y/z) [for sphere and cylinder]
```

as well as two other special variables

```
user user1            will monitor the [Mon_Name]_Vars.UserVariable{1|2}
user2 user3           to be assigned in an other component (see below)
```

To tell the component what you want to monitor, just add the variable names in the *options* parameter. The data will be sorted into *bins* cells (default is 20), between some default *limits*, that can also be set by user. The *auto* option will automatically determine what limits should be used to have a good sampling of signals.

Important options

Each monitoring records the flux (sum of weights p) versus the given variables, except if the `signal=<variable>` word is used in the `options`. The *cm2* option will ask to normalize the flux to the monitor section surface, and the `capture` option uses the gold foil integrated 'capture' flux weightening (up to the cadmium cut-off):

$$\Phi_c = \int_0^{0.5eV} \frac{d\Phi}{d\lambda} \frac{\lambda}{\lambda_{2200m/s}} d\lambda \tag{10.5}$$

The `auto` option is probably the most useful one: it asks the monitor to automatically determine the best limits for each variable, in order to obtain the most significant monitored histogram. This option should preceed each variable, or be located after all variables in which case they are all affected. On the other hand, one may manually set the limits with the `limits=[min max]` option. If no limits are set monitor_nd uses predefined limits that usually make sense for most neutron scattering simulations. Example: the default upper energy limit is 100 meV, but may be changed with an options string like `options="energy limits 0 200"`. Note that the limits also apply in list mode (see below).

The `log` and `abs` options should be positioned before each variable to specify logarithmic binning and absolute value respectively.

The *borders* option will monitor variables that are outside the limits. These values are then accumulated on the 'borders' of the signal.

The output files

By default, the file names will be the component name, followed by a time stamp and automatic extensions showing what was monitored (such as `MyMonitor.x`). You can also set the filename in *options* with the *file* keyword followed by the file name that you want. The extension will then be added if the name does not contain a dot (.). Finally, the *filename* parameter may also be used.

The output files format are standard 1D or 2D McStas detector files. The *no file* option will *inactivate* monitor, and make it a single 0D monitor detecting integrated flux and counts. The *verbose* option will display the nature of the monitor, and the names of the generated files.

The 2D output When you ask the **Monitor_nD** to monitor only two variables (e.g. *options* = "x y"), a single 2D file of intensity versus these two correlated variables will be created.

The 1D output The **Monitor_nD** can produce a set of 1D files, one for each monitored variable, when using 1 or more than 2 variables, or when specifying the *multiple* keyword option.

The List output The **Monitor_nD** can additionally produce a *list* of variable values for neutrons that pass into the monitor. This feature is additive to the 1D or 2D output. By default only 1000 events will be recorded in the file, but you can specify for instance "*list* 3000 neutrons" or "*list all* neutrons". This last option may require a lot of memory and generate huge files. Note that the limits to the measured parameters also apply in this mode. To exemplify, a monitor_nd instance with the option string `"list all v"` will *only* record those neutrons which have a velocity below 100000 m/s, whereas an instance with the option string `"list all vx vy vz 0 2000"` will record all neutrons with $|v_x, v_y| < 100$ m/s and $0 < v_z < 2000$ m/s. Thus, in this latter case, any neutron travelling in the negative z-direction will be disregarded.

Monitor equivalences

In the following table 10.9, we show how the Monitor_nD may substitute any other McStas monitor.

Usage examples

McStas monitor	Monitor_nD equivalent
Divergence_monitor	*options*="dx bins=*ndiv* limits=[$-\alpha/2\alpha/2$], lambda bins=*nlam* limits=[λ_0 λ_1] file=*file*"
DivLambda_monitor	*options*="dx bins=*nh* limits=[$-h_{max}/2h_{max}/2$], dy bins=*nv* limits=[$-v_{max}/2v_{max}/2$]" *filename*=*file*
DivPos_monitor	*options*="dx bins=*ndiv* limits=[$-\alpha/2\alpha/2$], x bins=*npos*" *xmin*=x_{min} *xmax*=x_{max}
E_monitor	*options*="energy bins=*nchan* limits=[$E_{min}E_{max}$]"
EPSD_monitor	*options*="energy bins=n_E limits=[$E_{min}E_{max}$], x bins=*nx*" *xmin*=x_{min} *xmax*=x_{max}
Hdiv_monitor	*options*="dx bins=*nh* limits=[$-h_{max}/2h_{max}/2$]" *filename*=*file*
L_monitor	*options*="lambda bins=*nh* limits=[$-\lambda_{max}/2\lambda_{max}/2$]" *filename*=*file*
Monitor_4PI	*options*="sphere"
Monitor	*options*="inactivate"
PSDcyl_monitor	*options*="theta bins=*nr*,y bins=*ny*, cylinder" *filename*=*file* *yheight*=*height* *xwidth*=2*radius
PSDlin_monitor	*options*="x bins=*nx*" *xmin*=x_{min} *xmax*=x_{max} *ymin*=y_{min} *ymax*=y_{max} *filename*=*file*
PSD_monitor_4PI	*options*="theta y, sphere"
PSD_monitor	*options*="x bins=*nx*, y bins=*ny*" *xmin*=x_{min} *xmax*=x_{max} *ymin*=y_{min} *ymax*=y_{max} *filename*=*file*
TOF_cylPSD_monitor	*options*="theta bins=n_ϕ, time bins=*nt* limits=[t_0, t_1], cylinder" *filename*=*file* *yheight*=*height* *xwidth*=2*radius
TOFLambda_monitor	*options*="lambda bins=n_λ limits=[λ_0 λ_1], time bins=*nt* limits=[t_0, t_1]" *filename*=*file*
TOFlog_mon	*options*="log time bins=*nt* limits=[t_0, t_1]"
TOF_monitor	*options*="time bins=*nt* limits=[t_0, t_1]"

Table 10.9.: Using Monitor_nD in place of other components. All limits specifications may be advantageously replaced by an *auto* word preceeding each monitored variable. Not all file and dimension specifications are indicated (e.g. filename, xmin, xmax, ymin, ymax).

```
COMPONENT MyMonitor = Monitor_nD(
     xmin = -0.1, xmax = 0.1,
     ymin = -0.1, ymax = 0.1,
     options = "energy auto limits")
```

will monitor the neutron energy in a single 1D file (a kind of E_monitor)

- `options = "banana, theta limits=[10,130], bins=120, y bins=30"`
  is a theta/height banana detector.
- `options = "banana, theta limits=[10,130], auto time"`
  is a theta/time-of-flight banana detector.
- `options="x bins=30 limits=[-0.05 0.05] ; y"`
  will set the monitor to look at x and y. For y, default bins (20) and limits values (monitor dimensions) are used.
- `options="x y, auto, all bins=30"`
  will determine itself the required limits for x and y.
- `options="multiple x bins=30, y limits=[-0.05 0.05], all auto"`
  will monitor the neutron x and y in two 1D files.
- `options="x y z kx ky kz, all auto"`
  will monitor each of theses variables in six 1D files.
- `options="x y z kx ky kz, list all, all auto"`
  will monitor all theses neutron variables in one long list, one row per neutron event.
- `options="multiple x y z kx ky kz, and list 2000, all auto"`
  will monitor all theses neutron variables in one list of 2000 events and in six 1D files.
- `options="signal=energy, x y"`
  is a PSD monitor recording the mean energy of the beam as a function of x and y.

Monitoring user variables

There are two ways to monitor any quantity with Monitor_nD. This may be e.g. the number of neutron bounces in a guide, or the wavevector and energy transfer at a sample. The only requirement is to define the `user1` (and optionally `user2,user3`) variables of a given Monitor_nD instance.

Directly setting the user variables (simple) The first method uses directly the `user1` and `username1` component parameters to directly transfer the value and label, such as in the following example:

```
TRACE
(...)
COMPONENT UserMonitor = Monitor_nD(
  user1     = log(t), username1="Log(time)",
  options   ="auto user1")
```

The values to assign to `user2` and `user3` must be global instrument variables, or a component output variables as in `user1=MC_GETPAR(some_comp, outpar)`. Similarly, the `user2,user3` and `username2,username3` parameters may be used to control the second and third user variable, to produce eventually 2D/3D user variable correlation data and custom event lists.

Indirectly setting the user variables (only for professionals) It is possible to control the user variables of a given Monitor_nD instance anywhere in the instrument description. This method requires more coding, but has the advantage that a variable may be defined to store the result of a computation locally, and then transfer it into the UserMonitor, all fitting in an EXTEND block.

This is performed in a 4 steps process:

1. Declare that you intend to monitor user variables in a Monitor_nD instance (defined in TRACE):

```
DECLARE
%{ (...)
  %include "monitor_nd-lib"
  MONND_DECLARE(UserMonitor); // will monitor custom things in
      UserMonitor
%}
```

2. Initialize the label of the user variable (optional):

```
INITIALIZE
%{
  (...)
  MONND_USER_TITLE(UserMonitor, 1, "Log(time)");
%}
```

   The value '1' could be '2' or '3' for the `user2,user3` variable.

3. Set the user variable value in a TRACE component EXTEND block:

```
TRACE
(...)
COMPONENT blah = blah_comp(...)
EXTEND
%{  // attach a value to user1 in UserMonitor, could be much more
    comlex here.
  MONND_USER_VALUE(UserMonitor, 1, log(t));
%}
(...)
```

4. Tell the Monitor_nD instance to record user variables:

```
TRACE
(...)
COMPONENT UserMonitor = Monitor_nD(options="auto user1")
(...)
```

Setting the user variable values may either make use of the neutron parameters (x,y,z, vx,vy,vz, t, sx,sy,sz, p), access the internal variables of the component that sets the user variables (in this example, those from the `blah` instance), access any component OUTPUT parameter using the `MC_GETPAR` C macro(see chapter C), or simply use a global instrument variable. Instrument parameters can not be used directly.

Example: Number of neutron bounces in a guide In the following example, we show how the number of bounces in a polygonal guide may be monitored. Let us have a guide made of many Guide_gravity instances. We declare a global simulation variable `nbounces`, set it to 0 for each neutron entering the guide, and sum-up all bounces from each section, accessing the `Gvars` OUTPUT variable of component Guide_gravity. Then we ask Monitor_nD to look at that value.

```
DECLARE
%{
  double nbounces;
%}
TRACE
(...)
COMPONENT Guide_in = Arm() AT (...)
EXTEND
%{
  nbounces = 0;
%}

COMPONENT Guide1 = Guide_gravity(...) AT (...) RELATIVE PREVIOUS
EXTEND
%{
  if (SCATTERED) nbounces += GVars.N_reflection[0];
%}
(... many guide instances, copy/paste and change names automatically ...)
COMPONENT COPY(Guide1) = COPY(Guide1) AT (...) RELATIVE PREVIOUS
EXTEND
%{
  if (SCATTERED) nbounces += GVars.N_reflection[0];
%}

// monitor nbounces
COMPONENT UserMonitor = Monitor_nD(
  user1=nbounces, username1="Number of bounces",
  options="auto user1") AT (...)
(...)
```

Monitoring neutron parameter correlations, PreMonitor_nD

The first imediate usage of the Monitor_nD component is when one requires to identify cross-correlations between some neutron parameters, e.g. position and divergence (*aka* phase-space diagram). This latter monitor would be merely obtained with:

```
options="x dx, auto", bins=30
```

This example records the correlation between position and divergence of neutrons at a given instrument location.

Name:	PreMonitor_nD
Author:	System, E. Farhi
Input parameters	comp
Optional parameters	
Notes	

But it is also possible to search for cross-correlation between two part of the instrument simulation. One example is the acceptance phase-diagram, which shows the neutron caracteristics at the input required to reach the end of the simulation. This *spatial* correlation may be revealed using the **PreMonitor_nD** component. This latter stores the neutron parameters at a given instrument location, to be used at an other Monitor_nD location for monitoring.

The only parameter of **PreMonitor_nD** is the name of the associated Monitor_nD instance, which should use the `premonitor` option, as in the following example:

```
COMPONENT CorrelationLocation = PreMonitor_nD(comp = CorrelationMonitor)
AT (...)

  (... e.g. a guide system )

COMPONENT CorrelationMonitor  = Monitor_nD(
    options="x dx, auto, all bins=30, premonitor")
AT (...)
```

which performs the same monitoring as the previous example, but with a spatial correlation constrain. Indeed, it records the position *vs* the divergence of neutrons at the correlation location, but only if they reach the monitoring position. All usual Monitor_nD variables may be used, except the user variables. These latter may be defined as described in section 10.8 in an EXTEND block.

10.9. The `Res_monitor` McStas Component

Monitor for resolution calculations

Identification

- **Author:** Kristian Nielsen
- **Origin:** Risoe
- **Date:** 1999

Description

A single detector/monitor, used together with the Res_sample component to compute instrument resolution functions. Outputs a list of neutron scattering events in the sample along with their intensities in the detector. The output file may be analyzed with the mcresplot front-end.

Example: Res_monitor(filename="Output.res", res_sample_comp="RSample", xmin=-0.1, xmax=0.1, ymin=-0.1, ymax=0.1)

Setting the monitor geometry. The optional parameter 'options' may be set as a string with the following keywords. Default is rectangular ('square'):

```
box                 Box of size xwidth, yheight, zdepth
cylinder            To get a cylindrical monitor (diameter is xwidth, height is yheight).
banana              Same as cylinder, without top/bottom, on restricted angular area
disk                Disk flat xy monitor. diameter is xwidth.
sphrere             To get a spherical monitor (e.g. a 4PI) (diameter is xwidth).
square              Square flat xy monitor (xwidth, yheight)
```

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
res_sample_comp	WITH quotes	Name of Res_sample component in the instrument definition	
filename	string	Name of output file. If unset, use automatic name	0
options	str	String that specifies the geometry of the monitor	0
xwidth	m	Width/diameter of detector	.1
yheight	m	Height of detector	.1
zdepth	m	Thichness of detector	0
radius	m	Radius of sphere/cylinder monitor	0

Name	Unit	Description	Default
xmin	m	Lower x bound of detector opening	0
xmax	m	Upper x bound of detector opening	0
ymin	m	Lower y bound of detector opening	0
ymax	m	Upper y bound of detector opening	0
zmin	m	Lower z bound of detector opening	0
zmax	m	Upper z bound of detector opening	0
bufsize	1	Number of events to store. Use 0 to store all	0
restore_neutron	1	If set, the monitor does not influence the neutron state	0
live_calc	1	If set, the monitor directly outputs the resolution matrix	1

Links

- Component source code found in file `Res_monitor.comp`.

10.10. Polarisation-sensitive monitors

See the Polarisation chapter (A) for the underlying formalism; these monitors measure the projection of the polarisation vector, either alone or resolved against wavelength/scattering angle.

10.11. The `Pol_monitor` McStas Component

Polarisation sensitive monitor.

Identification

- **Author:** Peter Christiansen
- **Origin:** Risoe
- **Date:** July 2006

Description

A square single monitor that measures the projection of the polarisation along a given normalized m-vector (mx, my, mz). The measured quantity is: sx*mx+sy*my+mz*sz

Example: Pol_monitor(xwidth=0.1, yheight=0.1, nchan=11, mx=0, my=1, mz=0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xwidth	m	Width of detector	0.1
yheight	m	Height of detector	0.1
restore_neutron	1	If set, the monitor does not influence the neutron state	0
mx	1	X-component of monitor vector (can be negative)	0
my	1	Y-component of monitor vector (can be negative)	0
mz	1	Z-component of monitor vector (can be negative)	0
nowritefile	1	If set, monitor will skip writing to disk	0

Links

- Component source code found in file `Pol_monitor.comp`.

10.12. The `PolLambda_monitor` McStas Component

Polarisation and wavelength sensitive monitor.

Identification

- **Author:** Peter Christiansen
- **Origin:** Risoe
- **Date:** July 2006

Description

A square single monitor that measures the projection of the polarisation along a given normalized m-vector (mx, my, mz) as a function of wavelength.

Example: Pollambda_monitor(Lmin=1, Lmax=20, nL=20, xwidth=0.1, yheight=0.1, npol=11, mx=0, my=1, mz=0, filename="pollambdaMon.data")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xwidth	m	Width of detector	0.1
yheight	m	Height of detector	0.1
nL	1	Number of bins in wavelength	20
npol	1	Number of bins in Pol	21
restore_neutron	1	If set, the monitor does not influence the neutron state	0
filename	string	Name of file in which to store the data	0
nowritefile	1	If set, monitor will skip writing to disk	0
mx	1	X-component of monitor vector (can be negative)	0
my	1	Y-component of monitor vector (can be negative)	0
mz	1	Z-component of monitor vector (can be negative)	0
Lmin	Å	Minimum wavelength detected	
Lmax	Å	Maximum wavelength detected	

Links

- Component source code found in file `PolLambda_monitor.comp`.

10.13. The `MeanPolLambda_monitor` McStas Component

Polarisation and wavelength sensitive monitor.

Identification

- **Author:** Peter Christiansen
- **Origin:** Risoe
- **Date:** July 2006

Description

A square single monitor that measures the MEAN projection of the polarisation along a given normalized m-vector (mx, my, mz) as a function of wavelength.

Example: MeanPollambda_monitor(xwidth=0.1, yheight=0.1, npol=11, my=1, filename="meanpollambdaMon.data")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xwidth	m	Width of detector	0.1
yheight	m	Height of detector	0.1
nL	1	Number of bins in wavelength	20
restore_neutron	1	If set, the monitor does not influence the neutron state	0
filename	string	Name of file in which to store the data	0
nowritefile	1	If set, monitor will skip writing to disk	0
mx	1	X-component of monitor vector (can be negative)	0
my	1	Y-component of monitor vector (can be negative)	0
mz	1	Z-component of monitor vector (can be negative)	0
Lmin	Å	Minimum wavelength detected	
Lmax	Å	Maximum wavelength detected	

Links

- Component source code found in file `MeanPolLambda_monitor.comp`.

10.14. The `PSD_monitor_4PI_spin` McStas Component

Spherical position-sensitive detector with spin weighting

Identification

- **Author:** Erik B Knudsen
- **Origin:** DTU
- **Date:** April 17, 2010

Description

An (n times m) pixel spherical PSD monitor using a cylindrical projection. Mostly for test and debugging purposes.
Example: PSD_monitor_4PI(radius=0.1, nx=90, ny=90, filename="Output.psd")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nx	1	Number of pixel columns	90
ny	1	Number of pixel rows	90
radius	m	Radius of detector	
filename	str	Name of file in which to store the detector image	0
restore_neutron	1	If set, the monitor does not influence the neutron state x	0
mx	1	x-component of spin reference-vector	0
my	1	y-component of spin reference-vector	1
mz	1	z-component of spin reference-vector	0

Links

- Component source code found in file `PSD_monitor_4PI_spin.comp`.

11. Special-purpose components

The chapter deals with components that are not easily included in any of the other chapters because of their special nature, but which are still part of the McStas system.

One part of these components deals with splitting simulations into two (or more) stages. For example, a guide system is often not changed much, and a long simulation of neutron rays "surviving" through the guide system could be reused for several simulations of the instrument back-end, speeding up the simulations by (typically) one or two orders of magnitude. As of McStas 3.x, the recommended components for this are **MCPL_input** and **MCPL_output**, which store and read back neutron rays using the portable, compressed, multi-code MCPL event format (with backends for McStas/McXtrace as well as several other Monte Carlo transport codes – MCNP, PHITS, Geant4, Vitess, SIMRES). The older, code-specific event components (**Virtual_input**/**Virtual_output** and similar MCNP/Tripoli4/Vitess-specific readers/writers) perform the same role but are now in the **obsolete** component category (see the Obsolete chapter) and are kept only for backward compatibility; new instrument work should prefer MCPL.

Progress_bar is a simulation utility that displays the simulation status, but assumes the form of a component.

Components for simulating instrument resolution functions (**Res_sample**, **TOF_Res_sample** and **Res_monitor**) are documented in the Samples and Monitors chapters respectively, matching their current component-library category (they predate the present category split and were historically documented here).

11.1. MCPL: splitting simulations via portable neutron event files

MCPL_input reads the events of an MCPL file as the starting rays of a simulation and sets the number of rays to the number of events in the file times `repeat_count`. As in **KDSource**, the resulting stream of rays is split into one contiguous chunk per MPI process. With `repeat_count`>1 the events are replayed, with the weights divided accordingly; every replay after the first is smeared in velocity, position and/or direction (`v_smear`, `pos_smear`, `dir_smear`), so at least one of these must be set. By default the file is shared between the MPI processes, so `-mpi=N` does not change the number of rays. Setting `replicate_across_processes=1` restores the behaviour from before Oct 2026: `repeat_count` is rounded up to a multiple of the number of MPI processes, so that each process runs through the whole file. The former **MCPL_input_once** is now an **MCPL_input** with `repeat_count=1` forced, in the obsolete category.

11.2. The `MCPL_input` McStas Component

Source-like component that reads neutron state parameters from an mcpl-file.

Identification

- **Author:** Erik B Knudsen, Gregory S Tucker and Peter Willendrup
- **Origin:** DTU Physics / European Spallation Source ERIC
- **Date:** Mar 2016, consolidated with MCPL_input_once Oct 2026

Description

Source-like component that reads neutron state parameters from a binary mcpl-file.

MCPL is short for Monte Carlo Particle List, and is a format for sharing events between e.g. MCNP(X), Geant4 and McStas.

The –ncount given on the commandline is overwritten by (#particles in the file) x repeat_count. Particles that are not neutrons, or are outside [Emin, Emax], are counted but ABSORBed.

The rays form a stream that runs through the file repeat_count times. As in KDSource, this stream is split into one contiguous chunk per MPI process: each process seeks once to its start and reads on, wrapping around to the start of the file (with preload=1 it loads only the part of the file its chunk covers). Every event is thus used repeat_count times across the whole run, and its weight is divided by repeat_count. All passes after the first are smeared using v_smear, pos_smear and dir_smear (the first pass is too if always_smear is set), so repeat_count $>$ 1 MUST be combined with at least one *_smear option; if all are 0, repeat_count is reset to 1.

With replicate_across_processes=0 (the default, as in MCPL_input_once) the file is shared between the MPI processes and –mpi=N does not change the number of rays.

replicate_across_processes=1 restores the pre-Oct 2026 MPI behaviour: repeat_count is rounded up to a multiple of the number of MPI processes, so that every process runs through the whole file at least once, and all but the first pass on the master are smeared (or, with all *_smear=0, are identical copies).

Example: MCPL_input(filename=voutput,verbose=1,repeat_count=1,v_smear=0.1,pos_smear=0.001,dir_sm

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename	str	Name of neutron mcpl file to read	0
polarisationuse		If !=0 read polarisation vectors from file	1
verbose		Print debugging information for the first 10 neutrons read (on the master process)	0
Emin	meV	Lower energy bound. Particles found in the MCPL-file below the limit are skipped	0
Emax	meV	Upper energy bound. Particles found in the MCPL-file above the limit are skipped	FLT_MAX
repeat_count	1	Replay the contents of the MCPL file this number of times, dividing the event weights by the same number. MUST be combined with _smear options!	1
v_smear	1	When replaying events, make a flat MC choice within v_smear*V around particle velocity V	0
pos_smear	m	When replaying events, make a flat MC choice of position within pos_smear around particle starting position	0
dir_smear	deg	When replaying events, make an MC choice of direction within dir_smear around particle direction	0
preload		Load particles during INITIALIZE. On GPU preload is forced	0
always_smear		If !=0, also smear the events on their first pass through the file	0
replicate_across_processes		If !=0, with MPI round repeat_count up to a multiple of the number of processes, so that each process runs through the whole file	0

Links

- Component source code found in file `MCPL_input.comp`.

11.3. The `MCPL_output` McStas Component

Detector-like component that writes neutron state parameters into an mcpl-format binary, virtual-source neutron file.

Identification

- **Author:** Erik B Knudsen
- **Origin:** DTU Physics
- **Date:** Mar 2016

Description

Detector-like component that writes neutron state parameters into an mcpl-format binary, virtual-source neutron file.

MCPL is short for Monte Carlo Particle List, and is a new format for sharing events between e.g. MCNP(X), Geant4 and McStas.

When used with MPI, the component will output #MPI nodes individual MCPL files that can be merged using the mcpltool.

MCPL_output allows a few flags to tweak the output files: 1. If use_polarisation is unset (default) the polarisation vector will not be stored (saving space) 2. If doubleprec is unset (default) data will be stored as 32 bit floating points, effectively cutting the output file size in half. 3. Extra information may be attached to each ray in the form of a userflag, a user-defined variable wich is packed into 32 bits. If the user variable does not fit in 32 bits the value will be truncated and likely garbage. If more than one variable is to be attached to each neutron this must be packed into the 32 bits.

These features are set this way to keep file sizes as manageable as possible.

Example: MCPL_output(filename="voutput", verbose=1, userflag="flag", userflag-comment="Neutron Id")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename	str	Name of neutron file to write. If not given, the component name will be used.	0

Name	Unit	Description	Default
weight_mode	1	weight_mode=1: record initial ray count in MCPL stat:sum entry and rescale particle weights. weight_mode=2: classical (deprecated) mode of outputting particle weights directly.	1
verbose	1	If 1) Print summary information for created MCPL file. 2) Also print summary of first 10 particles information stored in the MCPL file. >2) Also print information for first 10 particles as they are being stored by McStas	0
polarisationuse	1	Enable storing the polarisation state of the neutron.	0
doubleprec	1	Use double precision storage	0
userflag	1	Extra variable to attach to each neutron. The value of this variable will be packed into a 32 bit integer.	""
userflagcomment	str	String variable to describe the userflag. If this string is empty (the default) no userflags will be stored.	""
buffermax	1	Maximal number of events to save (<= MAXINT), GPU/OpenACC only	0

Links

- Component source code found in file `MCPL_output.comp`.

11.4. KDSource: kernel-density-estimated source from particle lists

11.5. The `KDSource` McStas Component

Distributional source, based on a MCPL-format particle list, on which the Kernel Density Estimation (KDE) method is applied.

Identification

- **Author:** O.I.Abbate, N.S.Schmidt, updates for McStas by T.Kittelmann and P.Willendrup
- **Origin:** Instituto Balseiro
- **Date:** Dec 2022 / May 2026

Description

Distributional source, based on a MCPL-format particle list, on which the Kernel Density Estimation (KDE) method is applied.

It allows sampling more particles than the number present in a virtual, previously generated virtual source, without repeating samples, controlled via the nloop input parameter.

To function, this component requires a KDSource installation v.2.0.2 or later, as distributed on conda-forge and pypi and included with (conda-based) McStas 3.7.0 or later.

As inputs the component needs access to all of: <ol> The original MPCL file Outputs of a KDSource-analyzed / optimized KDE source: <ul> An XML parameter-file containing the needed configuration A "bandwidth" file (source_bws)   For information on performing KDSource analysis, please refer to the KDSource example notebooks and the KDSource online documentation (links below).

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename	str	Name of the XML parameters file containing KDSource definition.	0
Emin	meV	Lower energy bound. Particles found in the MCPL-file below the limit are skipped.	0
Emax	meV	Upper energy bound. Particles found in the MCPL-file above the limit are skipped.	FLT_MAX

Name	Unit	Description	Default
nloop	int	Number of times to loop through the file.	1

Links

- Component source code found in file `KDSource.comp`.
- N.S. Schmidt et. al Annals of Nuclear Energy, Volume 177, 2022, 109309
- Test_KDSource example instrument
- KDSource online documentation (not fully updated)
- KDSource verification notebooks

11.6. Text: drawing a text label in mcdisplay

11.7. The `Text` McStas Component

Release: McStas 3.9
Draw a text

Identification

- **Author:** Emmanuel Farhi
- **Origin:** SOLEIL
- **Date:** Oct 10, 2026

Description

A text with given size and orientation. The text is drawn in upper letters.
Example: Text(text="SOLEIL", size=0.5, plane="XY")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
text	str	Text to draw.	
size	m	Size of text.	0.5
plane	str	"XY","YZ",or "XZ"	"XY"
pen_thickness	1	Stroke diameter as fraction of text size. 0 draws thin lines, >0 draws each stroke as a cylinder with spherical joints.	0.1

Links

- Component source code found in file `Text.comp`.

11.8. Progress_bar: Simulation progress and automatic saving

Name:	Progress_bar
Author:	System
Input parameters	percent, flag_save, profile
Optional parameters	
Notes	

This component displays the simulation progress and status but does not affect the neutron parameters. The display is updated in regular intervals of the full simulation; the default step size is 10 %, but it may be changed using the `percent` parameter (from 0 to 100). The estimated computation time is displayed at the begining and actual simulation time is shown at the end.

Additionally, setting the `flag_save` to 1 results in a regular save of the data files during the simulation. This means that is is possible to view the data before the end of the computation, and have also a trace of it in case of computer crash. The achieved percentage of the simulation is stored in these temporary data files. Technically, this save is equivalent to sending regularly a USR2 signal to the running simulation.

The optional 'profile' parameter, when set to a file name, will produce the number of statistical events reaching each component in the simulation. This may be used to identify positions where events are lost.

11.9. Beam_spy: A beam analyzer

Name:	Beam_spy
Author:	System
Input parameters	
Optional parameters	
Notes	should overlap previous component

Note: `Beam_spy` has moved to the **obsolete** component category (see the Component Manual's Obsolete chapter); the general-purpose `Monitor_nD` component (Monitors chapter) now covers this functionality and more. The description below is kept for reference.

This component is at the same time an Arm and a simple Monitor. It analyzes all neutrons reaching it, and computes statistics for the beam, as well as the intensity.

This component does not affect the neutron beam, and does not contain any propagation call. Thus it gets neutrons from the previous component in the instrument description, and should better be placed at the same position, with `AT (0,0,0) RELATIVE PREVIOUS`.

12. The Union framework: sample environments and multiple scattering

The Union framework, contributed by Mads Bertelsen (University of Copenhagen) [Ber17; BJL17], is a set of components that together allow the description of complex, possibly nested or overlapping, sample environments with multiple scattering, by separating the description of *geometry* (where matter is) from *physical process* (what happens to a neutron there). For a conceptual introduction to the framework – volumes, priority, masks, exit volumes, loggers, conditionals, tagging, and the underlying propagation algorithm – see chapter B. This chapter documents the input parameters of every individual Union component.

12.1. Union framework core: initialization, master, and termination

12.2. The `Union_init` McStas Component

Deprecated no-op placeholder, kept for backward compatibility.

Identification

- **Author:** Mads Bertelsen
- **Origin:** ESS DMSC
- **Date:** 20.08.15

Description

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas.

This component used to be required as the FIRST Union component. It defined the shared library (union-lib.c) and owned the shared global lists, which the other Union components reached via COMP_GETPAR3.

That is no longer needed: every Union component loads the Union library (union-lib) itself, and the library owns the shared lists. The Union_init and Union_stop components are therefore optional.

This definition is kept only so that existing instruments that still place a Union_init component continue to compile. It emits a deprecation warning and has no effect.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default

Links

- Component source code found in file `Union_init.comp`.
- **/

12.3. The `Union_master` McStas Component

Master component for Union components that performs the overall simulation

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) This master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

Algorithm: Described elsewhere

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
enable_refraction	0/1	Toggles refraction effects, simulated for all materials that have refraction parameters set	1
enable_reflection	0/1	Toggles reflection effects, simulated for all materials that have refraction parameters set	1
verbal	0/1	Toogles printing information loaded during initialize	0
list_verbal	0/1	Toogles information of all internal lists in intersection network	0
finally_verbal	0/1	Toogles information about cleanup performed in finally section	0
allow_inside_start	0/1	Set to 1 if rays are expected to start inside a volume in this master	0
enable_tagging	0/1	Enable tagging of ray history (geometry, scattering process)	0
history_limit	1	Limit the number of unique histories that are saved	300000
enable_conditionals	0/1	Use conditionals with this master	1
inherit_number_of_scattering_events	0/1	Inherit the number of scattering events from last master	0
weight_ratio_limit	1	Limit on the ratio between initial weight and current, ray absorbed if ratio becomes lower than limit	1e-90
display_mode	0/1/2	Volume display in mcdisplay: 0 follows –trace, 1 forces lines, 2 polyhedra (only with –trace=2, –trace=1 always gives lines)	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_master.comp`.

12.4. The `Union_master_GPU` McStas Component

Master component for Union components that performs the overall simulation

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) This master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

Algorithm: Described elsewhere

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
verbal	0/1	Toogles terminal output describing the defined simulation	1
list_verbal	0/1	Toogles information of all internal lists in intersection network	0
finally_verbal	0/1	Toogles information about cleanup performed in finally section	0
allow_inside_start	0/1	Set to 1 if rays are expected to start inside a volume in this master	0
enable_tagging	0/1	Enable tagging of ray history (geometry, scattering process)	0
history_limit	1	Limit the number of unique histories that are saved	300000
enable_conditionals	0/1	Use conditionals with this master	1
inherit_number_of_scattering_events	0/1	Inherit the number of scattering events from last master	0

Name	Unit	Description	Default
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_master_GPU.comp`.

12.5. The `Union_stop` McStas Component

Deprecated no-op placeholder, kept for backward compatibility.

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas.

This component used to be required as the LAST component. It pulled in the dispatch switch (union-suffix.c) so it would be compiled after all the process components had defined their PROCESS DETECTOR macros.

That is no longer needed: each process registers its own case with the dispatch in Union_master, whose TRACE code is compiled after every component SHARE section. The Union_init and Union_stop components are therefore optional.

This definition is kept only so that existing instruments that still place a Union_stop component continue to compile. It emits a deprecation warning and has no effect.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default

Links

- Component source code found in file `Union_stop.comp`.
- **/

12.6. Union geometry components

12.7. The `Union_box` McStas Component

A box geometry component for the Union components

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union components

The position of this component is the center of the box, zdepth/2 in each direction.

It is allowed to overlap components, but it is not allowed to have two parallel planes that coincides. This will crash the code on run time.

Algorithm: Described elsewhere

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
material_string	string	Material name of this volume, defined using Union_make_material	0

Name	Unit	Description	Default
priority	1	Priotiry of the volume (can not be the same as another volume) A high priority is on top of low.	
xwidth	m	Width of the box volume	
yheight	m	Height of the box volume	
zdepth	m	Depth of the box volume	
xwidth2	m	Optional different width at the +z box face	-1
yheight2	m	Optional different height at the +z box face	-1
visualize	1	Set to 0 if you wish to hide this geometry in mcdisplay	1
target_index	string	Focuses on component a component this many steps further in the component sequence	0
target_x	m	X position of target to focus at	0
target_y	m	Y position of target to focus at	0
target_z	m	Z position of target to focus at	0
focus_aw	deg	Horiz. angular dimension of a rectangular area	0
focus_ah	deg	Vert. angular dimension of a rectangular area	0
focus_xw	m	Horiz. dimension of a rectangular area	0
focus_xh	m	Vert. dimension of a rectangular area	0
focus_r	m	Focusing on circle with this radius	0
plus_z_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to +z face of box	0
minus_z_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to -z face of box	0
plus_x_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to +x face of box	0
minus_x_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to -x face of box	0
plus_y_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to +y face of box	0

Name	Unit	Description	Default
minus_y_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to -z face of box	0
all_face_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to all faces above	0
cut_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to all internal cuts	0
p_interact	1	Probability to interact with this geometry [0-1]	0
mask_string	string	Comma seperated list of geometry names which this geometry should mask	0
mask_setting	string	"All" or "Any", should the masked volume be simulated when the ray is in just one mask, or all.	0
number_of_activations	1	Number of subsequent Union_master components that will simulate this geometry	1
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile. CALCULATED PARAMETERS:	""

Links

- Component source code found in file `Union_box.comp`.

12.8. The `Union_cylinder` McStas Component

Cylinder geometry component for Union components

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

The position of this component is the center of the cylinder, and it thus extends yheight/2 up and down along y axis.

It is allowed to overlap components, but it is not allowed to have two parallel planes that coincides. This will crash the code on run time.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
material_string	string	Material name of this volume, defined using Union_make_material	0
priority	1	Priotiry of the volume (can not be the same as another volume) A high priority is on top of low.	
radius	m	Outer radius volume in (x,z) plane	
yheight	m	Cylinder height in (y) direction	
visualize	1	Set to 0 if you wish to hide this geometry in mcdisplay	1
target_index	1	Focuses on component a component this many steps further in the component sequence	0
target_x	m	X Position of target to focus at	0
target_y	m	Y Position of target to focus at	0
target_z	m	Z Position of target to focus at	0
focus_aw	deg	Horiz. angular dimension of a rectangular area	0
focus_ah	deg	Vert. angular dimension of a rectangular area	0
focus_xw	m	Horiz. dimension of a rectangular area	0
focus_xh	m	Vert. dimension of a rectangular area	0
focus_r	m	Focusing on circle with this radius	0

Name	Unit	Description	Default
p_interact	1	Probability to interact with this geometry [0-1]	0
mask_string	string	Comma seperated list of geometry names which this geometry should mask	0
mask_setting	string	"All" or "Any", should the masked volume be simulated when the ray is in just one mask, or all.	0
number_of_activations	1	Number of subsequent Union_master components that will simulate this geometry	1
curved_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to curved wall of cylinder	0
top_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to +y top face of cylinder	0
bottom_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to -y bottom face of cylinder	0
all_face_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to all faces above	0
cut_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to all internal cuts	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_cylinder.comp`.

12.9. The `Union_sphere` McStas Component

Sphere geometry component for the Union components

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union components

The position of this component is the center of the sphere

It is allowed to overlap components, but it is not allowed to have two parallel planes that coincides. This will crash the code on run time.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
material_string	string	material name of this volume, defined using Union_make_material	0
priority	1	priotiry of the volume (can not be the same as another volume) A high priority is on top of low.	
radius	m	Radius of sphere	
visualize	1	set to 0 if you wish to hide this geometry in mcdisplay	1
target_index	1	Focuses on component a component this many steps further in the component sequence	0
target_x	m	X position of target to focus at	0
target_y	m	Y position of target to focus at	0
target_z	m	Z position of target to focus at	0
focus_aw	deg	horiz. angular dimension of a rectangular area	0

Name	Unit	Description	Default
focus_ah	deg	vert. angular dimension of a rectangular area	0
focus_xw	m	horiz. dimension of a rectangular area	0
focus_xh	m	vert. dimension of a rectangular area	0
focus_r	m	focusing on circle with this radius	0
p_interact	1	probability to interact with this geometry [0-1]	0
mask_string	string	Comma seperated list of geometry names which this geometry should mask	0
mask_setting	string	"All" or "Any", should the masked volume be simulated when the ray is in just one mask, or all.	0
number_of_activations	1	Number of subsequent Union_master components that will simulate this geometry	1
surface	string	Comma seperated string of Union surface definitions defined prior to this component	0
cut_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to all internal cuts	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_sphere.comp`.

12.10. The `Union_cone` McStas Component

Cone geometry component for Union components

Identification

- **Author:** Martin Olsen
- **Origin:** University of Copenhagen
- **Date:** 17.09.18

Description

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

The position of this component is the center of the cone, and it thus extends yheight/2 up and down along y axis.

It is allowed to overlap components, but it is not allowed to have two parallel planes that coincides. This will crash the code on run time.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
material_string	string	Material name of this volume, defined using Union_make_material	0
priority	1	Priotiry of the volume (can not be the same as another volume) A high priority is on top of low.	
radius	m	Radius volume in (x,z) plane	0
radius_top	m	Top radius volume in (x,z) plane	0
radius_bottom	m	Bottom radius volume in (x,z) plane	0
yheight	m	Cone height in (y) direction	
visualize	1	Set to 0 if you wish to hide this geometry in mcdisplay	1
target_index	1	Focuses on component a component this many steps further in the component sequence	0
target_x	m	X position of target to focus at	0
target_y	m	Y position of target to focus at	0
target_z	m	Z position of target to focus at	0
focus_aw	deg	Horiz. angular dimension of a rectangular area	0
focus_ah	deg	Vert. angular dimension of a rectangular area	0
focus_xw	m	Horiz. dimension of a rectangular area	0

Name	Unit	Description	Default
focus_xh	m	Vert. dimension of a rectangular area	0
focus_r	m	Focusing on circle with this radius	0
p_interact	1	Probability to interact with this geometry [0-1]	0
mask_string	string	Comma seperated list of geometry names which this geometry should mask	0
mask_setting	string	"All" or "Any", should the masked volume be simulated when the ray is in just one mask, or all.	0
number_of_activations	1	Number of subsequent Union_master components that will simulate this geometry	1
curved_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to curved wall of cone	0
top_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to +y top face of cone	0
bottom_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to -y bottom face of cone	0
all_face_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to all faces above	0
cut_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to all internal cuts	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_cone.comp`.

12.11. The `Union_mesh` McStas Component

Component for including 3D mesh in Union geometry

Identification

- **Author:** Martin Olsen, and expanded upon by Daniel Lomholt Christensen
- **Origin:** University of Copenhagen / ACTNXT project
- **Date:** 17.09.18

Description

Part of the Union components, a set of components that work together and thus separates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere/mesh, assigned a material 4) A Union_master component placed after all the above

Only in step 4 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

The mesh component loads a 3D OFF, STL or PLY file as the geometry, using geometry-mesh-lib (polygons are triangulated, duplicated vertices welded). Faces given more than once are removed, so a volumetric mesh (cells sharing faces) gives its outer surface. The mesh geometry that is loaded must be a watertight mesh: every edge must be shared by exactly two triangles.

Number of intersections per ray: for each mesh, Union_master stores the GMESH_MAX_HITS (default 50) nearest crossings of the ray with the mesh surface. A concave mesh, or one made of several parts, can be crossed more often. Then the farther crossings are lost, the ray can not find the surface where it leaves the mesh, and Union_master reports trace errors. geometry-mesh-lib warns once when this happens: Warning: <file>: a ray crossed the mesh N times, but only the 4 nearest hits are kept To fix it, add this line after the DEFINE INSTRUMENT line of the instrument, with N at least the number given in the warning: DEPENDENCY " -DGMESH_MAX_HITS=N "

It is allowed to overlap components, but it is not allowed to have two parallel planes that coincides. This will crash the code on run time.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename	str	Name of file that contains the 3D geometry: OFF, STL (ascii or binary) or PLY	0
material_string	string	material name of this volume, defined using Union_make_material	0

Name	Unit	Description	Default
priority	1	priotiry of the volume (can not be the same as another volume) A high priority is on top of low.	
visualize	1	set to 0 if you wish to hide this geometry in mcdisplay	1
all_surfaces	string	Comma seperated string of Union surface definitions defined prior to this component	0
cut_surface	string	Comma seperated string of Union surface definitions defined prior to this component, applied to all internal cuts	0
target_index	1	Focuses on component a component this many steps further in the component sequence	0
target_x	m	position of target to focus at	0
target_y	m	Position of target to focus at	0
target_z	m	position of target to focus at	0
focus_aw	deg	horiz. angular dimension of a rectangular area	0
focus_ah	deg	vert. angular dimension of a rectangular area	0
focus_xw	m	horiz. dimension of a rectangular area	0
focus_xh	m	vert. dimension of a rectangular area	0
focus_r	m	focusing on circle with this radius	0
p_interact	1	probability to interact with this geometry [0-1]	0
mask_string	string	Comma seperated list of geometry names which this geometry should mask	0
mask_setting	string	"All" or "Any", should the masked volume be simulated when the ray is in just one mask, or all.	0
number_of_activations	1	Number of subsequent Union_master components that will simulate this geometry	1
coordinate_scale	1e-3	Multiplier that the vertex positions are multiplied by. Set to 1 for input coordinates in meters. Set to 1e-3 for input coordinates in mm.	1e-3

Name	Unit	Description	Default
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""
bvh_display_layers	1	mcdisplay draws the bounding boxes of this layer of the mesh BVH (the root is layer 1) when the BVH has more layers than this, otherwise the triangle edges. Set to 0 to always draw the triangle edges.	12

Links

- Component source code found in file `Union_mesh.comp`.

12.12. Union material definition

12.13. The `Union_make_material` McStas Component

Component that takes a number of Union processes and constructs a Union material

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

Part of the Union components, a set of components that work together and thus separates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using this component 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before the master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

Algorithm: Described elsewhere

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
process_string	string	Comma seperated names of physical processes	"NULL"
my_absorption	1/m	Inverse penetration depth from absorption at standard energy	
absorber	0/1	Control parameter, if set to 1 the material will have no scattering processes	0
refraction_density	g/cm^3	Density of the refracting material. density < 0 means the material is outside/before the shape.	0
refraction_sigma_coh	barn	Coherent cross section of refracting material. Use negative value to indicate a negative coherent scattering length	0
refraction_weight	g/mol	Molar mass of the refracting material	0
refraction_SLD	Å^{-2}	Scattering length density of material (overwrites sigma_coh, density and weight)	-1500
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_make_material.comp`.

12.14. The `Union_NCrystal_material` McStas Component

Version: $Revision: 0.1 $

Identification

- **Author:** Mads Bertelsen
- **Origin:** ESS DMSC
- **Date:** 2026-09-28

Description

Creates one NCrystal scattering process and a complete Union material. The absorption coefficient is obtained from the same NCrystal configuration, so density modifiers such as density=0.6x are applied automatically. Use NCrystal_process together with Union_make_material when NCrystal scattering must be combined with other Union processes. The user does not provide my_absorption for this component.

This component requires a Union geometry and Union_master.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
cfg	str	NCrystal material configuration string (details on this page).	""
interact_fraction	1	Fraction of scattering events assigned to this process 0-1 (default -1 lets Union assign it).	-1
include_SLD	0/1	Include the scattering length density calculated from cfg in the material refraction information.	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_NCrystal_material.comp`.

12.15. Union scattering process components

Additional Union processes, ported from MCViNE, are documented together with the other contributed components in section 14.112.

12.16. The `Non_process` McStas Component

This process dos nothing and is used for testing

Identification

- **Author:** Mads Bertelsen
- **Origin:** ESS DMSC
- **Date:** 20.08.15

Description

This process dos nothing and is used for testing

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components like this one 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box / Union_cylinder, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before the master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

Algorithm: Described elsewhere

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sigma	barns	Scattering cross section	5.08
packing_factor	1	How dense is the material compared to optimal 0-1	1
unit_cell_volume	$Å^3$	Unit cell volume	13.8
interact_fraction	1	How large a part of the scattering events should use this process 0-1 (sum of all processes in material = 1)	-1
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Non_process.comp`.

12.17. The `Incoherent_process` McStas Component

A sample component to separate geometry and phsysics

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

This Union_process is based on the Incoherent.comp component originally written by Kim Lefmann and Kristian Nielsen

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components like this one 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box / Union_cylinder, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before the master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

Algorithm: Described elsewhere

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sigma	barns	Incoherent scattering cross section	5.08
f_QE	1	Fraction of quasielastic scattering (rest is elastic) [1]	0
gamma	meV	Lorentzian width of quasielastic broadening (HWHM) [1]	0
packing_factor	1	How dense is the material compared to optimal 0-1	1
unit_cell_volume	$Å^3$	Unit cell volume	13.8
interact_fraction	1	How large a part of the scattering events should use this process 0-1 (sum of all processes in material = 1)	-1

Name	Unit	Description	Default
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Incoherent_process.comp`.
- The test/example instrument Test_Phonon.instr.

12.18. The `Inhomogenous_incoherent_process` McStas Component

A sample component to separate geometry and phsysics

Identification

- **Author:** Daniel Lomholt Christensen
- **Origin:** University of Copenhagen
- **Date:** 26/01/2026

Description

This Union_process is based on the Incoherent_process.comp component originally written by Mads Bertelsen inspired by Kim Lefmann and Kristian Nielsen

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components like this one 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box / Union_cylinder, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before the master, but after the previous will be simulated here.

There is a dedicated manual available for the Union components

Algorithm: The general algorithm for the Union system is described elsewhere.

I here give a brief introduction as to what changes occur when using an inhomogenous process in your Union make material. It is expected that you understand the basic algorithm of the Union system before reading this.

In Union, the neutron moves through a network of objects in a 3 dimensional world. When the neutron hits a material, the probability to scatter is calculated, and a Monte Carlo choice is taken, as to whether that neutron should scatter, or pass through. For a homogenous material (i.e constant attenuation coefficient μ), this probability is the Beer-Lambert law,

```
$P_s = 1 - e^{-\mu l}$
```

Where P_s is the scattering probability, and l is length of the neutron path throughout the object.

For an inhomogenous material, this Beer-Lambert law must be modified, as μ is a function of the position. Therefore the Beer-Lambert law becomes,

```
$P_s = \int^l_0 1 -  e^{-\mu(l')l'}dl'$
```

Calculating this μ in the inhomogenous case is often trivial, but not feasible, from a software development point of view (seeing as many different functions of μ might be wanted). Instead the inhomogenous processes performs an approximate integral, by evaluating μ at a number of points along the neutron path (This number is in fact number_of_sample_points).

For this incoherent process, the linear attenuation coefficient is,

 $\mu = pack/V_u * 100 * \sigma$

Where $pack$ is the packing factor of the material (defaults to 1), V_u is the Unit cell volume, and σ is the scattering cross section in barns. μ therefore has units of m^{-1}.

For this component each factor in the attenuation coefficient can be a "tiny expression". This means that it can be a mathematical equation such as $\sigma_{expr} = "5.08 + 1000 * z * 2.35"$. When the attenuation coefficient is calculated, then the current value of z is used to get σ.

The parameters that the tiny expression can rely upon are currently: The positions, x, y, z The velocities vx, vy, vz and the time t

McStas uses a sligthly modified version of tiny expressions that evaluate exponentials from right to left instead of the standard left to right. Furthermore McStas has added two functions to tiny expressions. These are: A heavy side function hvs(variable, switch_point, large_val,small_val) which returns large val if variable > switch_point and small val otherwise.

A gaussian distribution:
gauss(A,sig,x), which evaluates to A*1/sqrt(2*PI)/sig*exp(-x^2/2/sig^2)
An example using these can be found in the Test instrument for this component, called Test_inhomogenous_process.instr. Example #9 implements a gaussian and a heavyside function. For more information on tiny expressions, see the link below.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sigma	barns	Incoherent scattering cross section	0
sigma_expr	string	Tiny expression to be calculated as replacement for sigma	""
packing_factor	1	How dense is the material compared to optimal 0-1	1
packing_factor_expr	string	Tiny expression to be calculated as replacement for packing factor	""
unit_cell_volume	Å^3	Unit cell volume	0
unit_cell_volume_expr	string	Tiny expression to be calculated as replacement for the unit cell volume	""
gamma	meV	Lorentzian width of quasielastic broadening (HWHM) [1]	0
gamma_expr	meV	Tiny expression to be calculated as replacement for the gamma value.	""
f_QE	1	Fraction of quasielastic scattering (rest is elastic) [1]	0
number_of_sample_p	points	Number of points that are sampled along the neutron path through a material	20
interact_fraction	1	How large a part of the scattering events should use this process 0-1 (sum of all processes in material = 1)	-1
verbose	1	Flag that prints out the values calculated in the cross section calculation	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Inhomogenous_incoherent_process.comp`.

- For information on how to write a tiny expression, see their github repository

12.19. The `IncoherentPhonon_process` McStas Component

A component to simulate inelastic scattering in the incoherent approximation Takes into account one, two, and three phonon scattering explicitly, and multi-phonon scattering (with n>4) via the saddle point method

Identification

- **Author:** Victor Laliena
- **Origin:** University of Zaragoza
- **Date:** 06.11.2018

Description

Part of the Union components, a set of components that work together and thus expects geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components like this one 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box / Union_cylinder, assigned a material 4) A Union_master component placed after all of the above

Algorithm: Described elsewhere, see e.g. https://doi.org/10.3233/JNR-190117

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
T	K	Temperature	394
density	g/cm^3	Material density	6.0
M	amu	ion mass	50.94
sigmaCoh	barns	Coherent scattering cross section	0.0184
sigmaInc	barns	Incoherent scattering cross section	5.08
DOSfn	string	Path to the file that contains the DoS	NULL
nphe_exact	1	Number of terms in the phonon expansion taken exact. Has to be between 1 and 3.	1
nphe_approx	1	Number of terms in the phonon expansion taken approximate.	0

Name	Unit	Description	Default
approx	1	Approximation type: 0 gaussian, 1 saddle point	0
mph_resum	0/1	Resumate the remaining terms of the phonon expansion via a saddle point: 0 No, 1 Yes	0
nxs	1	Number of energy points at which the total cross sections are precomputed	1000
kabsmin	A^{-1}	Lower cut-off for the neutron wave-vector k	0.1
kabsmax	A^{-1}	Higher cut-off for the neutron wave-vector k	25
interact_fraction	1	How large a part of the scattering events should use this process 0-1 (sum of all processes in material = 1)	-1
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `IncoherentPhonon_process.comp`.
- See https://doi.org/10.3233/JNR-190117

12.20. The `PhononSimple_process` McStas Component

Port of PhononSimple to Union components

Identification

- **Author:** Anders Komar Ravn, based on template by Mads Bertelsen and Phonon_Simple
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

Port of the PhononSimple component from the McStas library to the Union components.

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components like this one 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box / Union_cylinder, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before the master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

Algorithm: Described elsewhere

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
packing_factor	1	How dense is the material compared to optimal 0-1	1
unit_cell_volume	Å^3	Unit cell volume	13.8
interact_fraction	1	How large a part of the scattering events should use this process 0-1 (sum of all processes in material = 1)	-1
a	Å	fcc lattice constant	4.95
c	meV*Å	Velocity of sound	10
M	amu	Atomic mass	207.2
b	fm	Scattring length	9.4
T	K	Temperature	290
DW	1	Debye-Waller factor	1
longitudinal	0/1	Simulate longitudinal branches	1
transverse	0/1	Simulate transverse branches	1
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile. CALCULATED PARAMETERS: PhononSimple_storage // Important to update this output paramter effective_my_scattering // Variable used in initialize	""

Links

- Component source code found in file `PhononSimple_process.comp`.

12.21. The `Powder_process` McStas Component

Port of the PowderN process to the Union components

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

This Union_process is based on the PowderN.comp component originally written by P. Willendrup, L. Chapon, K. Lefmann, A.B.Abrahamsen, N.B.Christensen, E.M.Lauridsen.

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components like this one 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box / Union_cylinder, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before the master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components Algorithm: Described elsewhere

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
reflections	string	Input file for reflections. No scattering if NULL or "" [string]	"NULL"
packing_factor	1	How dense is the material compared to optimal 0-1	1
Vc	Å^3	Volume of unit cell=nb atoms per cell/density of atoms.	0
delta_d_d	0/1	Global relative delta_d_d/d broadening when the 'w' column is not available. Use 0 if ideal.	0
DW	1	Global Debye-Waller factor when the 'DW' column is not available. Use 1 if included in F2	0

Name	Unit	Description	Default
nb_atoms	1	Number of sub-unit per unit cell, that is ratio of sigma for chemical formula to sigma per unit cell	1
d_phi	deg	Angle corresponding to the vertical angular range to focus to, e.g. detector height. 0 for no focusing.	0
density	g/cm^3	Density of material. rho=density/weight/1e24*N_A.	0
weight	g/mol	Atomic/molecular weight of material.	0
barns	1	Flag to indicate if |F|^2 from 'reflections' is in barns or fm^2 (barns=1 for laz, barns=0 for lau type files).	1
Strain	ppm	Global relative delta_d_d/d shift when the 'Strain' column is not available. Use 0 if ideal.	0
interact_fraction	1	How large a part of the scattering events should use this process 0-1 (sum of all processes in material = 1)	-1
format	no quotes	Name of the format, or list of column indexes (see Description).	{0, 0, 0, 0, 0, 0, 0, 0, 0, 0}

Links

- Component source code found in file `Powder_process.comp`.

12.22. The `Single_crystal_process` McStas Component

Port of the Single_crystal component to the Union components

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

This Union_process is based on the Single_crystal.comp component originally written by Kristian Nielsen

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components like this one 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box / Union_cylinder, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before the master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

Algorithm: Described elsewhere

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
reflections	string	File name containing structure factors of reflections. Use empty ("") or NULL for incoherent scattering only	0
delta_d_d	1	Lattice spacing variance, gaussian RMS	1e-4
mosaic	arc minutes	Crystal mosaic (isotropic), gaussian RMS. Puts the crystal in the isotropic mosaic model state, thus disregarding other mosaicity parameters.	-1
mosaic_a	arc minutes	Horizontal (rotation around lattice vector a) mosaic (anisotropic), gaussian RMS. Put the crystal in the anisotropic crystal vector state. I.e. model mosaicity through rotation around the crystal lattice vectors. Has precedence over in-plane mosaic model.	-1
mosaic_b	arc minutes	Vertical (rotation around lattice vector b) mosaic (anisotropic), gaussian RMS.	-1
mosaic_c	arc minutes	Out-of-plane (Rotation around lattice vector c) mosaic (anisotropic), gaussian RMS	-1

Name	Unit	Description	Default
mosaic_AB	arc_minutes, arc_minutes, 1, 1, 1, 1, 1, 1	In Plane mosaic rotation and plane vectors (anisotropic), mosaic_A, mosaic_B, A_h,A_k,A_l, B_h,B_k,B_l. Puts the crystal in the in-plane mosaic state. Vectors A and B define plane in which the crystal roation is defined, and mosaic_A, mosaic_B, denotes the resp. mosaicities (gaussian RMS) with respect to the two reflections chosen by A and B (Miller indices).	{0,0, 0,0,0, 0,0,0}
recip_cell	1	Choice of direct/reciprocal (0/1) unit cell definition	0
barns	1	Flag to indicate if |F|^2 from 'reflections' is in barns or fm^2. barns=1 for laz and isotropic constant elastic scattering (reflections=NULL), barns=0 for lau type files	0
ax	Å or $Å^{-1}$	Coordinates of first (direct/recip) unit cell vector	0
ay	Å or $Å^{-1}$	a on y axis	0
az	Å or $Å^{-1}$	a on z axis	0
bx	Å or $Å^{-1}$	Coordinates of second (direct/recip) unit cell vector	0
by	Å or $Å^{-1}$	b on y axis	0
bz	Å or $Å^{-1}$	b on z axis	0
cx	Å or $Å^{-1}$	Coordinates of third (direct/recip) unit cell vector	0
cy	Å or $Å^{-1}$	c on y axis	0
cz	Å or $Å^{-1}$	c on z axis	0
aa	deg	Unit cell angles alpha, beta and gamma. Then uses norms of vectors a,b and c as lattice parameters	0
bb	deg	Beta angle	0
cc	deg	Gamma angle	0
order	1	Limit multiple scattering up to given order (0: all, 1: first, 2: second, ...) (Not supported in Union)	0
RX	m	Radius of lattice curvature along X. flat when 0.	0
RY	m	Radius of lattice curvature along Y. flat when 0.	0

Name	Unit	Description	Default
RZ	m	Radius of lattice curvature along Z. flat when 0.	0
powder	1	Flag to indicate powder mode, for simulation of Debye-Scherrer cones via random crystallite orientation. A powder texture can be approximated with 0	0
PG	1	Flag to indicate "Pyrolytic Graphite" mode, only meaningful with choice of Graphite.lau, models PG crystal. A powder texture can be approximated with 0	0
interact_fraction	1	How large a part of the scattering events should use this process 0-1 (sum of all processes in material = 1)	-1
packing_factor	1	How dense is the material compared to optimal 0-1	1

Links

- Component source code found in file `Single_crystal_process.comp`.

12.23. The `AF_HB_1D_process` McStas Component

1D Antiferromagnetic Heisenberg chain

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

1D Antiferromagnetic Heisenberg chain

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components like this one 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box / Union_cylinder, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before the master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

Algorithm: Described elsewhere

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
atom_distance	Å	Distance between atom's in chain	1
number_density	$1/Å^3$	Number of scatteres per volume	0
unit_cell_volume	$Å^3$	Unit cell volume (set either unit_cell_volume or number density)	0
A_constant	unitless	Constant from Müller paper 1981, probably somewhere between 1 and 1.5	1
J_interaction	meV	Exchange constant	1
packing_factor	1	How dense is the material compared to optimal 0-1	1
interact_fraction	1	How large a part of the scattering events should use this process 0-1 (sum of all processes in material = 1)	-1
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `AF_HB_1D_process.comp`.

12.24. The `Texture_process` McStas Component

Component for simulating textured powder in Union components

Identification

- **Author:** Victor Laliena
- **Origin:** University of Zaragoza
- **Date:** 2018-2019

Description

Part of the Union components, a set of components that work together and thus seperates geometry and physics within McStas. The use of this component requires other components to be used.

This component deals with the coherent elastic scattering on a textured material. The texture is described through the coefficients of the generalized Fourier transform of the Orientation Distribution Function (ODF). The component expects as input two files, one containing the Fourier coefficients of the ODF and another one with crystallographic and physical information.

Algorithm: Described elsewhere, see e.g. https://doi.org/10.3233/JNR-190117

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
crystal_fn	s	Path to a file (.laz) containing crystallographic and physical information	0
fcoef_fn	s	Path to a text file containing the Fourier coefficients of the ODF. It must have five columns: l m n Real(Clmn) Imag(Clmn)	0
lmax_user	1	Cut-off for the Fourier series. If negative, the program uses all the coefficients provided in the file fcoef_fn	-1
barns	1	Flag to indicate if |F|^2 from 'crystal_fn' is in barns or fm^2. barns=1 for laz and isotropic constant elastic scattering (reflections=NULL), barns=0 for lau type files	0
order	1	Limit scattering to this order	0
interact_fraction	1	How large a part of the scattering events should use this process 0-1 (sum of all processes in material = 1)	-1
packing_factor	1	How dense is the material compared to optimal 0-1	1
maxNeutronSaved	1	Maximum number of neutron cross sections saved for reusing	1
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Texture_process.comp`.
- See https://doi.org/10.3233/JNR-190117

12.25. The `NCrystal_process` McStas Component

This file is part of NCrystal (see https://mctools.github.io/ncrystal/).

Copyright 2015-2022 NCrystal developers

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. McStas, neutron ray-tracing package Copyright(C) 2007 Risoe National Laboratory.

Original header text for NCrystal_sample.comp: McStas sample component for the NCrystal scattering library. Find more information at <a href="https://github.com/mctools/ncrystal NCrystal wiki. In particular, browse the available datafiles at Data-library and read about format of the configuration string expected in the "cfg" parameter at Using-NCrystal.

NCrystal is available under the Apache 2.0 license. Depending on the configuration choices, optional NCrystal modules under different licenses might be enabled - see <a href="https://github.com/mctools/ncry for more details.

Identification

- **Author:** NCrystal developers, converted to a Union component by Mads Bertelsen
- **Origin:** NCrystal Developers (European Spallation Source ERIC and DTU Nutech)
- **Date:** 20.08.15

Description

This process uses the NCrystal library as a Union scattering process. It only provides physics; geometry and absorption are handled by the other Union components. Use Union_make_material to provide the absorption coefficient. Use Union_NCrystal_material for a complete material backed by one NCrystal configuration.

Part of the Union components, a set of components that work together and thus separates geometry and physics within McStas. The use of this component requires other components to be used:

1) One specifies a number of processes using process components like this one 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box / Union_cylinder, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before the master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components.

Algorithm: Described elsewhere

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
cfg	str	NCrystal material configuration string (details on this page).	""
interact_fraction	1	How large a part of the scattering events should use this process 0-1 (sum of all processes in material = 1)	-1
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `NCrystal_process.comp`.
- CALCULATED PARAMETERS:
- NCrystal_storage // Storage used to communicate with the probability and scattering function
- **/

12.26. The `Template_process` McStas Component

A template process for building Union processes

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

This is a template for a new contributor to create their own physical process. The comments in this file are meant to teach the user about creating their own process file, rather than explaining this one. For comments on how this code works, look in the Incoherent_process.comp.

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components like this one 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box / Union_cylinder, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before the master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

Algorithm: Described elsewhere

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sigma	barns	Incoherent scattering cross section	5.08
packing_factor	1	How dense is the material compared to optimal 0-1	1
unit_cell_volume	Å^3	Unit_cell_volume	13.8
interact_fraction	1	How large a part of the scattering events should use this process 0-1 (sum of all processes in material = 1)	-1
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Template_process.comp`.

12.27. The `Template_surface` McStas Component

A template process for building Union surface processes

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

This is a template for a new contributor to create their own surface process. The comments in this file are meant to teach the user about creating their own surface component, rather than explaining this one. For comments on how this code works, look in the Mirror_surface.comp. To add a new surface process, three changes are needed in union-lib.h: Add entry in surface enum (match your process) Add a pointer to your storage struct in surface_data_transfer_union Add an empty UNION_CASE_PHYSICS_SURFACE_<NAME> default and list it in UNION_CASES_PHYSICS_SURFACE; your SHARE section then defines the case

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components like this one 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box / Union_cylinder, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before the master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

Algorithm: Described elsewhere

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
reflect	str	Name of reflectivity file. Format q(Å-1) R(0-1)	0

Name	Unit	Description	Default
R0	1	Low-angle reflectivity	0.99
Qc	$Å^{-1}$	Critical scattering vector	0.0219
alpha	Å	Slope of reflectivity	6.07
m	1	m-value of material. Zero means completely absorbing.	2
W	$Å^{-1}$	Width of supermirror cut-off	0.003
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Template_surface.comp`.

12.28. The `Mirror_surface` McStas Component

A Mirror surface process

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

This is a Union surface process that describes a supermirror or other surface that only have specular reflection. The reflectivity can be given as a file or using the standard reflectivity inputs. To use this in a simulation an instance of this component should be defined in the instrument file, then attatched to one or more geometries in their surface stacks pertaining to each face of the geometry.

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components like this one 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box / Union_cylinder, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before the master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

Algorithm: Described elsewhere

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
reflect	str	Name of reflectivity file. Format q(Å-1) R(0-1)	0
R0	1	Low-angle reflectivity	0.99
Qc	$Å^{-1}$	Critical scattering vector	0.0219
alpha	Å	Slope of reflectivity	6.07
m	1	m-value of material. Zero means completely absorbing.	2
W	$Å^{-1}$	Width of supermirror cut-off	0.003
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Mirror_surface.comp`.

12.29. Union loggers (event/history recording)

12.30. The `Union_logger_1D` McStas Component

One dimensional Union logger for several possible variables

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

This logger in particular is not finished. It is supposed to allow several different variables to be histogrammed, but currently only time is supported

A logger will log something for scattering events happening to certain volumes, which are specified in the target_geometry string. By leaving it blank, all geometries are logged, even the ones not defined at this point in the instrument file. If a list og target_geometries is selected, one can further narrow the events logged by providing a list of process names in target_process which need to correspond with names of defined Union_process components.

To use the logger_conditional_extend function, set it to some integer value n and make and extend section to the master component that runs the geometry. In this extend function, logger_conditional_extend[n] is 1 if the conditional stack evaluated to true, 0 if not. This way one can check what rays is logged using regular McStas monitors. Only works if a conditional is applied to this logger.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_geometry	string	Comma seperated list of geometry names that will be logged, leave empty for all volumes (even not defined yet)	"NULL"
target_process	string	Comma seperated names of physical processes, if volumes are selected, one can select Union_process names	"NULL"
min_value	1	Histogram boundery for logged value	
max_value	1	Histogram boundery for logged value	
n1	1	Number of bins in histogram	90
variable	string	Time for time in seconds, q for magnitude of scattering vector in 1/Å.	"time"
filename	string	Filename of produced data file	"NULL"

Name	Unit	Description	Default
order_total	1	Only log rays that scatter for the n'th time, 0 for all orders	0
order_volume	1	Only log rays that scatter for the n'th time in the same geometry	0
order_volume_process	1	Only log rays that scatter for the n'th time in the same geometry, using the same process	0
logger_conditional_extend_index	1	If a conditional is used with this logger, the result of each conditional calculation can be made available in extend as a array called "logger_conditional_extend", and one would then acces logger_conditional_extend[n] if logger_conditional_extend_index is set to n	-1
nowritefile	1	If set, logger will skip writing to disk	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_logger_1D.comp`.

12.31. The `Union_logger_2DQ` McStas Component

Two dimensional logger of scattering vector for Union components

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

This logger logs a 2D projection of the scattering vector, q in the lab frame.

A logger will log something for scattering events happening to certain volumes, which are specified in the target_geometry string. By leaving it blank, all geometries are logged, even the ones not defined at this point in the instrument file. If a list og target_geometries is selected, one can further narrow the events logged by providing a list of process names in target_process which need to correspond with names of defined Union_process components.

To use the logger_conditional_extend function, set it to some integer value n and make and extend section to the master component that runs the geometry. In this extend function, logger_conditional_extend[n] is 1 if the conditional stack evaluated to true, 0 if not. This way one can check what rays is logged using regular McStas monitors. Only works if a conditional is applied to this logger.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_geometry	string	Comma seperated list of geometry names that will be logged, leave empty for all volumes (even not defined yet)	"NULL"
target_process	string	Comma seperated names of physical processes, if volumes are selected, one can select Union_process names	"NULL"
Q_direction_1	string	Q direction for first axis ("x", "y" or "z")	"x"
Q1_min	$Å^{-1}$	Histogram boundery, min q value for first axis	-5
Q1_max	$Å^{-1}$	Histogram boundery, max q value for first axis	5
n1	1	Number of bins for first axis	90
Q_direction_2	string	Q direction for second axis ("x", "y" or "z")	"z"
Q2_min	$Å^{-1}$	Histogram boundery, min q value for second axis	-5
Q2_max	$Å^{-1}$	Histogram boundery, max q value for second axis	5

Name	Unit	Description	Default
n2	1	Number of bins for second axis	90
filename	string	Filename of produced data file	"NULL"
order_total	1	Only log rays that scatter for the n'th time, 0 for all orders	0
order_volume	1	Only log rays that scatter for the n'th time in the same geometry	0
order_volume_process	1	Only log rays that scatter for the n'th time in the same geometry, uwsing the same process	0
logger_conditional_extend_index	1	If a conditional is used with this logger, the result of each conditional calculation can be made available in extend as a array called "logger_conditional_extend", and one would then acces logger_conditional_extend[n] if logger_conditional_extend_index is set to n	-1
nowritefile	1	If set, logger will skip writing to disk	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_logger_2DQ.comp`.

12.32. The `Union_logger_2D_kf` McStas Component

Two dimensional logger of final wavevector for Union components

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

This logger logs a 2D projection of the final wavevector after each scattering in the lab frame.

A logger will log something for scattering events happening to certain volumes, which are specified in the target_geometry string. By leaving it blank, all geometries are logged, even the ones not defined at this point in the instrument file. If a list og target_geometries is selected, one can further narrow the events logged by providing a list of process names in target_process which need to correspond with names of defined Union_process components.

To use the logger_conditional_extend function, set it to some integer value n and make and extend section to the master component that runs the geometry. In this extend function, logger_conditional_extend[n] is 1 if the conditional stack evaluated to true, 0 if not. This way one can check what rays is logged using regular McStas monitors. Only works if a conditional is applied to this logger.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_geometry	string	Comma seperated list of geometry names that will be logged, leave empty for all volumes (even not defined yet)	"NULL"
target_process	string	Comma seperated names of physical processes, if volumes are selected, one can select Union_process names	"NULL"
Q1_min	$Å^{-1}$	Histogram boundery, min kf value for first axis	-5
Q1_max	$Å^{-1}$	Histogram boundery, max kf value for first axis	5
Q2_min	$Å^{-1}$	Histogram boundery, min kf value for second axis	-5

Name	Unit	Description	Default
Q2_max	$Å^{-1}$	Histogram boundery, max kf value for second axis	5
Q_direction_1	string	kf direction for first axis ("x", "y" or "z")	"x"
Q_direction_2	string	kf direction for second axis ("x", "y" or "z")	"z"
filename	string	Filename of produced data file	"NULL"
n1	1	Number of bins for first axis	90
n2	1	Number of bins for second axis	90
order_total	1	Only log rays that scatter for the n'th time, 0 for all orders	0
order_volume	1	Only log rays that scatter for the n'th time in the same geometry	0
order_volume_process	1	Only log rays that scatter for the n'th time in the same geometry, using the same process	0
logger_conditional_extend_index	1	If a conditional is used with this logger, the result of each conditional calculation can be made available in extend as a array called "logger_conditional_extend", and one would then acces logger_conditional_extend[n] if logger_conditional_extend_index is set to n	-1
nowritefile	1	If set, logger will skip writing to disk	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_logger_2D_kf.comp`.

12.33. The `Union_logger_2D_kf_time` McStas Component

Two dimensional logger of wavevector for a range of time bins

Identification

- **Author:** Mads Bertelsen

- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

This logger logs a 2D projection of the final wavevector after each scattering in the lab frame. It will do so for a number of time_bins, making it possible to make a animation, but beware of memory / diskspace requirements.

A logger will log something for scattering events happening to certain volumes, which are specified in the target_geometry string. By leaving it blank, all geometries are logged, even the ones not defined at this point in the instrument file. If a list og target_geometries is selected, one can further narrow the events logged by providing a list of process names in target_process which need to correspond with names of defined Union_process components.

To use the logger_conditional_extend function, set it to some integer value n and make and extend section to the master component that runs the geometry. In this extend function, logger_conditional_extend[n] is 1 if the conditional stack evaluated to true, 0 if not. This way one can check what rays is logged using regular McStas monitors. Only works if a conditional is applied to this logger.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_geometry	string	Comma seperated list of geometry names that will be logged, leave empty for all volumes (even not defined yet)	"NULL"
target_process	string	Comma seperated names of physical processes, if volumes are selected, one can select Union_process names	"NULL"
Q1_min	A^{-1}	Histogram boundery, min kf value for first axis	-5

Name	Unit	Description	Default
Q1_max	A^{-1}	Histogram boundery, max kf value for first axis	5
Q2_min	A^{-1}	Histogram boundery, min kf value for second axis	-5
Q2_max	A^{-1}	Histogram boundery, max kf value for second axis	5
time_min	s	Minimum time	0
time_max	s	Maximum time	1
Q_direction_1	string	kf direction for first axis ("x", "y" or "z")	"x"
Q_direction_2	string	kf direction for second axis ("x", "y" or "z")	"z"
filename	string	Filename of produced data file	"NULL"
n1	1	Number of bins for first axis	90
n2	1	Number of bins for second axis	90
time_bins	1	Number of time bins	10
order_total	1	Only log rays that scatter for the n'th time, 0 for all orders	0
order_volume	1	Only log rays that scatter for the n'th time in the same geometry	0
order_volume_process	1	Only log rays that scatter for the n'th time in the same geometry, using the same process	0
logger_conditional_extend_index	1	If a conditional is used with this logger, the result of each conditional calculation can be made available in extend as a array called "logger_conditional_extend", and one would then acces logger_conditional_extend[n] if logger_conditional_extend_index is set to n	-1
nowritefile	1	If set, logger will skip writing to disk	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_logger_2D_kf_time.comp`.

12.34. The `Union_logger_2D_space` McStas Component

Two dimensional spatial logger for the Union components

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

This logger logs a 2D projection of the position of each scattering in the lab frame.

This logger needs to be placed in space, the position is the center of the histogram.

A logger will log something for scattering events happening to certain volumes, which are specified in the target_geometry string. By leaving it blank, all geometries are logged, even the ones not defined at this point in the instrument file. If a list og target_geometries is selected, one can further narrow the events logged by providing a list of process names in target_process which need to correspond with names of defined Union_process components.

To use the logger_conditional_extend function, set it to some integer value n and make and extend section to the master component that runs the geometry. In this extend function, logger_conditional_extend[n] is 1 if the conditional stack evaluated to true, 0 if not. This way one can check what rays is logged using regular McStas monitors. Only works if a conditional is applied to this logger.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_geometry	string	Comma seperated list of geometry names that will be logged, leave empty for all volumes (even not defined yet)	"NULL"
target_process	string	Comma seperated names of physical processes, if volumes are selected, one can select Union_process names	"NULL"
D_direction_1	string	Direction for first axis ("x", "y" or "z")	"x"
D1_min	m	Histogram boundery, min position value for first axis	-5
D1_max	m	Histogram boundery, max position value for first axis	5
n1	1	Number of bins for first axis	90
D_direction_2	string	Direction for second axis ("x", "y" or "z")	"z"
D2_min	m	Histogram boundery, min position value for second axis	-5
D2_max	m	Histogram boundery, max position value for second axis	5
n2	1	Number of bins for second axis	90
filename	string	Filename of produced data file	"NULL"
order_total	1	Only log rays that scatter for the n'th time, 0 for all orders	0
order_volume	1	Only log rays that scatter for the n'th time in the same geometry	0
order_volume_process	1	Only log rays that scatter for the n'th time in the same geometry, using the same process	0
logger_conditional_extend_index	1	If a conditional is used with this logger, the result of each conditional calculation can be made available in extend as a array called "logger_conditional_extend", and one would then acces logger_conditional_extend[n] if logger_conditional_extend_index is set to n	-1
nowritefile	1	If set, logger will skip writing to disk	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_logger_2D_space.comp`.

12.35. The `Union_logger_2D_space_time` McStas Component

Two dimensional spatail logger for a number of time bins

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

This logger logs a 2D projection of the position of each scattering in the lab frame. Using the time_bins one can select to get this project for different time slots, effectively making a small animation of what happens.

A logger will log something for scattering events happening to certain volumes, which are specified in the target_geometry string. By leaving it blank, all geometries are logged, even the ones not defined at this point in the instrument file. If a list og target_geometries is selected, one can further narrow the events logged by providing a list of process names in target_process which need to correspond with names of defined Union_process components.

To use the logger_conditional_extend function, set it to some integer value n and make and extend section to the master component that runs the geometry. In this extend function, logger_conditional_extend[n] is 1 if the conditional stack evaluated to true, 0 if not. This way one can check what rays is logged using regular McStas monitors. Only works if a conditional is applied to this logger.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_geometry	string	Comma seperated list of geometry names that will be logged, leave empty for all volumes (even not defined yet)	"NULL"
target_process	string	Comma seperated names of physical processes, if volumes are selected, one can select Union_process names	"NULL"
D1_min	m	Histogram boundery, min position value for first axis	-5
D1_max	m	Histogram boundery, max position value for first axis	5
D2_min	m	Histogram boundery, min position value for second axis	-5
D2_max	m	Histogram boundery, max position value for second axis	5
time_min	s	Minimum time	0
time_max	s	Maximum time	1
D_direction_1	string	Direction for first axis ("x", "y" or "z")	"x"
D_direction_2	string	Direction for second axis ("x", "y" or "z")	"z"
filename	string	Filename of produced data file	"NULL"
n1	1	Number of bins for first axis	90
n2	1	Number of bins for second axis	90
time_bins	1	Number of time bins	10
order_total	1	Only log rays that scatter for the n'th time, 0 for all orders	0
order_volume	1	Only log rays that scatter for the n'th time in the same geometry	0
order_volume_process	1	Only log rays that scatter for the n'th time in the same geometry, using the same process	0
logger_conditional_extend_index	1	If a conditional is used with this logger, the result of each conditional calculation can be made available in extend as a array called "logger_conditional_extend", and one would then acces logger_conditional_extend[n] if logger_conditional_extend_index is set to n	-1
nowritefile	1	If set, logger will skip writing to disk	0

Name	Unit	Description	Default
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_logger_2D_space_time.comp`.

12.36. The `Union_logger_3D_space` McStas Component

Three dimensional logger for spatial dimensions

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

This logger logs a 2D projection of the position of each scattering in the lab frame. It does this in n3 space slices, so one can get a full 3D histogram of scattering positions.

A logger will log something for scattering events happening to certain volumes, which are specified in the target_geometry string. By leaving it blank, all geometries are logged, even the ones not defined at this point in the instrument file. If a list og target_geometries is selected, one can further narrow the events logged by providing a list of process names in target_process which need to correspond with names of defined Union_process components.

To use the logger_conditional_extend function, set it to some integer value n and make and extend section to the master component that runs the geometry. In this extend function, logger_conditional_extend[n] is 1 if the conditional stack evaluated to true, 0 if not. This way one can check what rays is logged using regular McStas monitors. Only works if a conditional is applied to this logger.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_geometry	string	Comma seperated list of geometry names that will be logged, leave empty for all volumes (even not defined yet)	"NULL"
target_process	string	Comma seperated names of physical processes, if volumes are selected, one can select Union_process names	"NULL"
D1_min	m	histogram boundery, min position value for first axis	-1
D1_max	m	histogram boundery, max position value for first axis	1
D2_min	m	histogram boundery, min position value for second axis	-1
D2_max	m	histogram boundery, max position value for second axis	1
D3_min	m	histogram boundery, min position value for third axis	-1
D3_max	m	histogram boundery, max position value for third axis	1
D_direction_1	string	Direction for first axis ("x", "y" or "z")	"x"
D_direction_2	string	Direction for second axis ("x", "y" or "z")	"z"
D_direction_3	string	Direction for second axis ("x", "y" or "z")	"z"
filename	string	Filename of produced data file	"NULL"
n1	1	number of bins for first axis	90
n2	1	number of bins for second axis	90
n3	1	number of bins for third axis	10
order_total	1	Only log rays that scatter for the n'th time, 0 for all orders	0
order_volume	1	Only log rays that scatter for the n'th time in the same geometry	0

Name	Unit	Description	Default
order_volume_process	1	Only log rays that scatter for the n'th time in the same geometry, using the same process	0
logger_conditional_extend_index	1	If a conditional is used with this logger, the result of each conditional calculation can be made available in extend as a array called "logger_conditional_extend", and one would then acces logger_conditional_extend[n] if logger_conditional_extend_index is set to n	-1
nowritefile	1	If set, logger will skip writing to disk	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_logger_3D_space.comp`.

12.37. Union absorption loggers

12.38. The `Union_abs_logger_1D_space` McStas Component

Logger of absorption along 1D space

Identification

- **Author:** Mads Bertelsen
- **Origin:** ESS DMSC
- **Date:** 19.06.20

Description

Part of the Union components, a set of components that work together and thus separates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using

Union_box/cylinder/sphere, assigned a material 4) Logger and conditional components can be placed which will record what happens 5) A Union_master component placed after all of the above

Only in step 5 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

This component is an absorption logger, and thus placed in point 4) above.

A absorption logger will log something for each absorption event happening in the geometry or geometries on which it is attached. These are specified in the target_geometry string. By leaving it blank, all geometries are logged, even the ones not defined at this point in the instrument file. Multiple geometries are specified as a comma separated list.

This absorption logger stores the absorbed weight as a function of height in a histogram. It is expected this abs logger will often be attached to a cylindrical geometry, and so if the abs logger has the same position and orientation it will measure absorption along the axis of symmetry. This makes it easy to create simple tubes, and it is even possible to include the detector casing and similar to improve the realism of the simulation.

This absorption logger needs to be placed in space, the position is recorded in the coordinate system of the logger component.

It is possible to attach one or more conditional components to this absorption logger. Such a conditional component would impose a condition on the state of the neutron after the Union_master component that executes the simulation, and the absorption logger will only record the event if this condition is true.

To use the logger_conditional_extend function, set it to some integer value n and make and extend section to the master component that runs the geometry. In this extend function, logger_conditional_extend[n] is 1 if the conditional stack evaluated to true, 0 if not. This way one can check what rays is logged using regular McStas monitors. Only works if a conditional is applied to this logger.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_geometry	string	Comma separated list of geometry names that will be logged, leave empty for all volumes (even not defined yet)	"NULL"
yheight	m	height of absorption logger	
n	1	number of bins along y	100
filename	string	Filename of produced data file	"NULL"
order_total	1	Only log rays that have scattered n times, -1 for all orders	-1
order_volume	1	Only log rays that have scattered n times in the same geometry, -1 for all orders	-1

Name	Unit	Description	Default
logger_conditional_extend_index	1	If a conditional is used with this logger, the result of each conditional calculation can be made available in extend as a array called "logger_conditional_extend", and one would then access logger_conditional_extend[n] if logger_conditional_extend_index is set to n	-1
nowritefile	1	If set, logger will skip writing to disk	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_abs_logger_1D_space.comp`.

12.39. The `Union_abs_logger_1D_space_event` McStas Component

Logger of absorption in 1D space stored as events

Identification

- **Author:** Mads Bertelsen
- **Origin:** ESS DMSC
- **Date:** 19.06.20

Description

Part of the Union components, a set of components that work together and thus separates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) Logger and conditional components can be placed which will record what happens 5) A Union_master component placed after all of the above

Only in step 5 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

This component is an absorption logger, and thus placed in point 4) above.

A absorption logger will log something for each absorption event happening in the geometry or geometries on which it is attached. These are specified in the target_geometry string. By leaving it blank, all geometries are logged, even the ones not defined at this point in the instrument file. Multiple geometries are specified as a comma separated list.

This absorption logger stores the absorbed weight as an event including a pixel_id. The pixel_id is found from the y coordinate of the event position in the coordinate system of the absorption logger and is binned to a pixel_id which is necessary when importing the data in Mantid. The y coordinate corresponds to the height and is natural when attaching this absorption logger to a cylindrical geometry, as it will record position along the axis of symmetry, and thus behave like a detector tube. When using several of these absorption loggers, they will internally avoid reusing the same pixel_id, but if any other mantid detectors are used, these need to have pixel_ids larger than the maximum used by these loggers. Each of these components uses three times the number of bins, as it internally is a 3xn histogram where only the center line have data, as this is much easier to import in Mantid. The ocmponent uses Monitor_nD functions for trace and mcdisplay, and for this reason it can write the xml file required to transfer the detector geometry to Mantid.

This absorption logger needs to be placed in space, the position is recorded in the coordinate system of the logger component.

It is possible to attach one or more conditional components to this absorption logger. Such a conditional component would impose a condition on the state of the neutron after the Union_master component that executes the simulation, and the absorption logger will only record the event if this condition is true.

To use the logger_conditional_extend function, set it to some integer value n and make and extend section to the master component that runs the geometry. In this extend function, logger_conditional_extend[n] is 1 if the conditional stack evaluated to true, 0 if not. This way one can check what rays is logged using regular McStas monitors. Only works if a conditional is applied to this logger.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_geometry	string	Comma separated list of geometry names that will be logged, leave empty for all volumes (even not defined yet)	"NULL"
yheight	m	Height of absorption logger	
n	1	Number of bins along y	

Name	Unit	Description	Default
fake_event	1	Number of fake events to add, circumvents issue with empty event lists with MPI	0
filename	string	Filename of produced data file	"NULL"
order_total	1	Only log rays that have scattered n times, -1 for all orders	-1
order_volume	1	Only log rays that have scattered n times in the same geometry, -1 for all orders	-1
logger_conditional_extend_index	1	If a conditional is used with this logger, the result of each conditional calculation can be made available in extend as a array called "logger_conditional_extend", and one would then access logger_conditional_extend[n] if logger_conditional_extend_index is set to n	-1
nowritefile	1	If set, logger will skip writing to disk	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_abs_logger_1D_space_event.comp`.

12.40. The `Union_abs_logger_1D_space_tof` McStas Component

Logger of absorption along 1D space and 1D time of flight

Identification

- **Author:** Mads Bertelsen
- **Origin:** ESS DMSC
- **Date:** 19.06.20

Description

Part of the Union components, a set of components that work together and thus separates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) Logger and conditional components can be placed which will record what happens 5) A Union_master component placed after all of the above

Only in step 5 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

This component is an absorption logger, and thus placed in point 4) above.

A absorption logger will log something for each absorption event happening in the geometry or geometries on which it is attached. These are specified in the target_geometry string. By leaving it blank, all geometries are logged, even the ones not defined at this point in the instrument file. Multiple geometries are specified as a comma separated list.

This absorption logger records the absorbed intensity as a function of the y position and time of flight. One common use could be to attach this absorption logger to a cylindrical helium-3 volume, creating a model of the detector volume. In such a detector, only the y position of the event is known, and as such creates a similar dataset.

This absorption logger needs to be placed in space, the position is recorded in the coordinate system of the logger component. Note the detection is along the y axis of the component, so it is natural to place it relative to a cylinder.

It is possible to attach one or more conditional components to this absorption logger. Such a conditional component would impose a condition on the state of the neutron after the Union_master component that executes the simulation, and the absorption logger will only record the event if this condition is true.

To use the logger_conditional_extend function, set it to some integer value n and make and extend section to the master component that runs the geometry. In this extend function, logger_conditional_extend[n] is 1 if the conditional stack evaluated to true, 0 if not. This way one can check what rays is logged using regular McStas monitors. Only works if a conditional is applied to this logger.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_geometry	string	Comma separated list of geometry names that will be logged, leave empty for all volumes (even not defined yet)	"NULL"
yheight	m	height of absorption logger	

Name	Unit	Description	Default
n	1	number of bins for spatial axis	0.2
time_min	s	Minimum time recorded	0
time_max	s	Maximum time recorded	1
time_bins	1	number of time bins	100
filename	string	Filename of produced data file	"NULL"
order_total	1	Only log rays that have scattered n times, -1 for all orders	-1
order_volume	1	Only log rays that have scattered n times in the same geometry, -1 for all orders	-1
logger_conditional_extend_index		If a conditional is used with this logger, the result of each conditional calculation can be made available in extend as a array called "logger_conditional_extend", and one would then access logger_conditional_extend[n] if logger_conditional_extend_index is set to n	-1
nowritefile	1	If set, logger will skip writing to disk	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_abs_logger_1D_space_tof.comp`.

12.41. The `Union_abs_logger_1D_space_tof_to_lambda` McStas Component

A logger of absorption along 1D of space and 1D of tof, but converted to lambda

Identification

- **Author:** Mads Bertelsen
- **Origin:** ESS DMSC
- **Date:** 19.06.20

Description

Part of the Union components, a set of components that work together and thus separates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) Logger and conditional components can be placed which will record what happens 5) A Union_master component placed after all of the above

Only in step 5 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

This component is an absorption logger, and thus placed in point 4) above.

A absorption logger will log something for each absorption event happening in the geometry or geometries on which it is attached. These are specified in the target_geometry string. By leaving it blank, all geometries are logged, even the ones not defined at this point in the instrument file. Multiple geometries are specified as a comma separated list.

This absorption logger records the absorbed intensity as a function of the measured and true wavelength. The measured wavelength is calculated from the time of flight and distance travelled. This distance is a constant from source to sample added to the distance from the sample position to the detector pixel in which the event is detected. The true wavelength is calculated directly from the velocity. This information shows any error in conversion from tof to wavelength, especially from any added travelled distance from multiple scattering.

The lambda_min, max and bin parameters are used to set both the range for measured and true wavelength. If either of these, denoted lambda_m and lambda_t respectively, is set they will overwrite the lambda setting for that part. Sine most data will be along the line lambda_m = lambda_t, it is possible to record lambda_m / lambda_t as a function of lambda_t, which will be close to 1.0. This mode is selected by setting relative_measured to 1, and then the range can be selected with relative_min, max and bins.

This absorption logger needs to be placed in space, the position is recorded in the coordinate system of the logger component. Note the detection is along the y axis of the component, so it is natural to place it relative to a cylinder.

This component works as other absorption loggers, but converts the tof and position data to wavelength, which is then compared to the actual wavelength calculated from the neutron state. The neutron position used to calculate the wavelength is pixelated while the time of flight is continous. The distance from source to sample is an input parameter, and the distance from sample to a detector pixel is calculated using the reference component position which should be specified with a relative component index.

It is possible to attach one or more conditional components to this absorption logger. Such a conditional component would impose a condition on the state of the neutron after the Union_master component that executes the simulation, and the absorption logger

will only record the event if this condition is true.

To use the logger_conditional_extend function, set it to some integer value n and make and extend section to the master component that runs the geometry. In this extend function, logger_conditional_extend[n] is 1 if the conditional stack evaluated to true, 0 if not. This way one can check what rays is logged using regular McStas monitors. Only works if a conditional is applied to this logger.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_geometry	string	Comma separated list of geometry names that will be logged, leave empty for all volumes (even not defined yet)	"NULL"
ref_component	1	Reference component instance index, sample position for distance calculation	
yheight	m	Height of absorption logger	
yn	1	Number of bins along y axis	
source_sample_dist	m	Travel distance between source and sample position, used to calculate total travelled distance	0.0
lambda_min	Å	Minimum wavelength recorded (sets both lambda_m_min and lambda_t_min)	-1
lambda_max	Å	Maximum wavelength recorded (sets both lambda_m_max and lambda_t_max)	-1
lambda_bins	1	Number of wavelength bins	-1
lambda_m_min	Å	Minimum measured wavelength recorded from tof and travelled distance (overwrites lambda_min)	-1
lambda_m_max	Å	Maximum measured wavelength recorded from tof and travelled distance (overwrites lambda_max)	-1
lambda_m_bins	1	Number of measured wavelength bins	-1
lambda_t_min	Å	Minimum true wavelength recorded from tof and travelled distance (overwrites lambda_min)	-1
lambda_t_max	Å	Maximum true wavelength recorded from tof and travelled distance (overwrites lambda_max)	-1
lambda_t_bins	1	Number of true wavelength bins	-1

Name	Unit	Description	Default
relative_measured	1	Default 0, records measured as function of true wavelength, if this is enabled, records measured relative to true wavelength	0
relative_min	1	Smallest value of measured / true wavelength in histogram	0.5
relative_max	1	Largest value of measured / true wavelength in histogram	1.5
relative_bins	1	Number of bins on histogram axis with measured divided by true wavelength	100
filename	string	Filename of produced data file	"NULL"
order_total	1	Only log rays that have scattered n times, -1 for all orders	-1
order_volume	1	Only log rays that have scattered n times in the same geometry, -1 for all orders	-1
logger_conditional_extend_index	1	If a conditional is used with this logger, the result of each conditional calculation can be made available in extend as a array called "logger_conditional_extend", and one would then access logger_conditional_extend[n] if logger_conditional_extend_index is set to n	-1
nowritefile	1	If set, logger will skip writing to disk	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_abs_logger_1D_space_tof_to_lambda.comp`.

12.42. The `Union_abs_logger_1D_time` McStas Component

Logger of absorption along 1D time

Identification

- **Author:** Mads Bertelsen

- **Origin:** ESS DMSC
- **Date:** 19.06.20

Description

Part of the Union components, a set of components that work together and thus separates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) Logger and conditional components can be placed which will record what happens 5) A Union_master component placed after all of the above

Only in step 5 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

This component is an absorption logger, and thus placed in point 4) above.

A absorption logger will log something for each absorption event happening in the geometry or geometries on which it is attached. These are specified in the target_geometry string. By leaving it blank, all geometries are logged, even the ones not defined at this point in the instrument file. Multiple geometries are specified as a comma separated list.

This absorption logger stores the absorbed weight as a function of height in a histogram. It is expected this abs logger will often be attached to a cylindrical geometry, and so if the abs logger has the same position and orientation it will measure absorption along the axis of symmetry. This makes it easy to create simple tubes, and it is even possible to include the detector casing and similar to improve the realism of the simulation.

This absorption logger needs to be placed in space, the position is recorded in the coordinate system of the logger component.

It is possible to attach one or more conditional components to this absorption logger. Such a conditional component would impose a condition on the state of the neutron after the Union_master component that executes the simulation, and the absorption logger will only record the event if this condition is true.

To use the logger_conditional_extend function, set it to some integer value n and make and extend section to the master component that runs the geometry. In this extend function, logger_conditional_extend[n] is 1 if the conditional stack evaluated to true, 0 if not. This way one can check what rays is logged using regular McStas monitors. Only works if a conditional is applied to this logger.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_geometry	string	Comma separated list of geometry names that will be logged, leave empty for all volumes (even not defined yet)	"NULL"
time_min	s	time at start of recording	0
time_max	s	Maximum time to log	
n	1	number of bins along y	100
filename	string	Filename of produced data file	"NULL"
order_total	1	Only log rays that have scattered n times, -1 for all orders	-1
order_volume	1	Only log rays that have scattered n times in the same geometry, -1 for all orders	-1
logger_conditional_extend_index	1	If a conditional is used with this logger, the result of each conditional calculation can be made available in extend as a array called "logger_conditional_extend", and one would then access logger_conditional_extend[n] if logger_conditional_extend_index is set to n	-1
nowritefile	1	If set, logger will skip writing to disk	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_abs_logger_1D_time.comp`.

12.43. The `Union_abs_logger_2D_space` McStas Component

Logger of absorption along two dimensions of space

Identification

- **Author:** Mads Bertelsen
- **Origin:** ESS DMSC
- **Date:** 19.06.20

Description

Part of the Union components, a set of components that work together and thus separates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) Logger and conditional components can be placed which will record what happens 5) A Union_master component placed after all of the above

Only in step 5 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

This component is an absorption logger, and thus placed in point 4) above.

A absorption logger will log something for each absorption event happening in the geometry or geometries on which it is attached. These are specified in the target_geometry string. By leaving it blank, all geometries are logged, even the ones not defined at this point in the instrument file. Multiple geometries are specified as a comma separated list.

This absorption logger stores the absorbed weight in a histogram spanning two spatial directions. The position of the event is projected onto this plane. The plotted data is very useful when ensuring the simulated geometry matches the intentions. It is not recommended to use this tool for safety concerns, as for example to estimate activity of a sample after irradiation.

The spatial plane in which the histogram is performed are chosen with the D_direction_1 and D_direction_2 parameters which can be "x", "y" or "z". The D1_min and D1_max parameters sets the limits for the first axis, and D2_min / D2_max likewise for the second axis.

This absorption logger needs to be placed in space, the position is recorded in the coordinate system of the logger component.

It is possible to attach one or more conditional components to this absorption logger. Such a conditional component would impose a condition on the state of the neutron after the Union_master component that executes the simulation, and the absorption logger will only record the event if this condition is true.

To use the logger_conditional_extend function, set it to some integer value n and make and extend section to the master component that runs the geometry. In this extend function, logger_conditional_extend[n] is 1 if the conditional stack evaluated to true, 0 if not. This way one can check what rays is logged using regular McStas monitors. Only works if a conditional is applied to this logger.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_geometry	string	Comma separated list of geometry names that will be logged, leave empty for all volumes (even not defined yet)	"NULL"
D_direction_1	string	Direction for first axis ("x", "y" or "z")	"x"
D1_min	m	Histogram boundary, min position value for first axis	-0.2
D1_max	m	Histogram boundary, max position value for first axis	5
n1	1	Number of bins for first axis	0.2
D_direction_2	string	Direction for second axis ("x", "y" or "z")	"z"
D2_min	m	Histogram boundary, min position value for second axis	-0.2
D2_max	m	Histogram boundary, max position value for second axis	5
n2	1	Number of bins for second axis	0.2
filename	string	Filename of produced data file	"NULL"
order_total	1	Only log rays that have scattered n times, -1 for all orders	-1
order_volume	1	Only log rays that have scattered n times in the same geometry, -1 for all orders	-1
logger_conditional_extend_index	1	If a conditional is used with this logger, the result of each conditional calculation can be made available in extend as a array called "logger_conditional_extend", and one would then access logger_conditional_extend[n] if logger_conditional_extend_index is set to n	-1
nowritefile	1	If set, logger will skip writing to disk	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_abs_logger_2D_space.comp`.

12.44. The `Union_abs_logger_event` McStas Component

A logger of absorption as events

Identification

- **Author:** Mads Bertelsen
- **Origin:** ESS DMSC
- **Date:** 19.06.20

Description

Part of the Union components, a set of components that work together and thus separates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) Logger and conditional components can be placed which will record what happens 5) A Union_master component placed after all of the above

Only in step 5 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

This component is an absorption logger, and thus placed in point 4) above.

A absorption logger will log something for each absorption event happening in the geometry or geometries on which it is attached. These are specified in the target_geometry string. By leaving it blank, all geometries are logged, even the ones not defined at this point in the instrument file. Multiple geometries are specified as a comma separated list.

This absorption logger stores absorption as events, with position, velocity, time and weight. The Monitor_nD libraries are used to write the event files.

This absorption logger needs to be placed in space, the position and velocity is recorded in the coordinate system of the logger component.

It is possible to attach one or more conditional components to this absorption logger. Such a conditional component would impose a condition on the state of the neutron after the Union_master component that executes the simulation, and the absorption logger will only record the event if this condition is true.

To use the logger_conditional_extend function, set it to some integer value n and make and extend section to the master component that runs the geometry. In this extend function, logger_conditional_extend[n] is 1 if the conditional stack evaluated to true, 0 if not. This way one can check what rays is logged using regular McStas monitors. Only works if a conditional is applied to this logger.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_geometry	string	Comma separated list of geometry names that will be logged, leave empty for all volumes (even not defined yet)	"NULL"
filename	string	Filename of produced data file	"NULL"
xwidth	m	Width in x direction	0
xbins	1	Number of bins in x direction	0
yheight	m	Height in y direction	0
ybins	1	Number of bins in y direction	0
zdepth	m	Depth in z direction	0
zbins	1	Number of bins in z direction	0
order_total	1	Only log rays that have scattered n times, -1 for all orders	-1
order_volume	1	Only log rays that have scattered n times in the same geometry, -1 for all orders	-1
logger_conditional_extend_index	1	If a conditional is used with this logger, the result of each conditional calculation can be made available in extend as a array called "logger_conditional_extend", and one would then access logger_conditional_extend[n] if logger_conditional_extend_index is set to n	-1
nowritefile	1	If set, logger will skip writing to disk	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_abs_logger_event.comp`.

12.45. The `Union_abs_logger_nD` McStas Component

A logger of absorption as events with Monitor_nD options

Identification

- **Author:** Mads Bertelsen
- **Origin:** ESS DMSC
- **Date:** 19.06.20

Description

Part of the Union components, a set of components that work together and thus separates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) Logger and conditional components can be placed which will record what happens 5) A Union_master component placed after all of the above

Only in step 5 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

This component is an absorption logger, and thus placed in point 4) above.

An absorption logger will log something for each absorption event happening in the geometry or geometries on which it is attached. These are specified in the target_geometry string. By leaving it blank, all geometries are logged, even the ones not defined at this point in the instrument file. Multiple geometries are specified as a comma separated list.

This absorption logger stores absorption as events, with position, velocity, time and weight. The Monitor_nD libraries are used to write the event files. This version is a close copy of Monitor_nD, having the same interface, though the user must be aware that no propagation happens for rays to hit the detector pixels, instead it uses the position of absorbed. Use the previous keyword to tell Monitor_nD that this is going on. It still needs values set for xwidth and yheight, even though these will not be used.

This absorption logger needs to be placed in space, the position and velocity is recorded in the coordinate system of the logger component.

It is possible to attach one or more conditional components to this absorption logger. Such a conditional component would impose a condition on the state of the neutron after the Union_master component that executes the simulation, and the absorption logger will only record the event if this condition is true.

To use the logger_conditional_extend function, set it to some integer value n and make and extend section to the master component that runs the geometry. In this extend function, logger_conditional_extend[n] is 1 if the conditional stack evaluated to true, 0 if not. This way one can check what rays is logged using regular McStas monitors. Only works if a conditional is applied to this logger.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_geometry	string	Comma separated list of geometry names that will be logged, leave empty for all volumes (even not defined yet)	"NULL"
order_total	1	Only log rays that have scattered n times, -1 for all orders	-1
order_volume	1	Only log rays that have scattered n times in the same geometry, -1 for all orders	-1
logger_conditional_extend_index	1	If a conditional is used with this logger, the result of each conditional calculation can be made available in extend as a array called "logger_conditional_extend", and one would then access logger_conditional_extend[n] if logger_conditional_extend_index is set to n	-1
user0	str	Variable name of USERVAR to be monitored by user0.	""
user1	str	Variable name of USERVAR to be monitored by user1.	""
user2	str	Variable name of USERVAR to be monitored by user2.	""
user3	str	Variable name of USERVAR to be monitored by user3.	""
user4	str	Variable name of USERVAR to be monitored by user4.	""
user5	str	Variable name of USERVAR to be monitored by user5.	""
user6	str	Variable name of USERVAR to be monitored by user6.	""
user7	str	Variable name of USERVAR to be monitored by user7.	""
user8	str	Variable name of USERVAR to be monitored by user8.	""
user9	str	Variable name of USERVAR to be monitored by user9.	""
xwidth	m	Width of detector.	0
yheight	m	Height of detector.	0
zdepth	m	Thickness of detector (z).	0
xmin	m	Lower x bound of opening	0

Name	Unit	Description	Default
xmax	m	Upper x bound of opening	0
ymin	m	Lower y bound of opening	0
ymax	m	Upper y bound of opening	0
zmin	m	Lower z bound of opening	0
zmax	m	Upper z bound of opening	0
bins	1	Number of bins to force for all variables. Use 'bins' keyword in 'options' for heterogeneous bins	0
min	u	Minimum range value to force for all variables. Use 'min' or 'limits' keyword in 'options' for other limits	-1e40
max	u	Maximum range value to force for all variables. Use 'max' or 'limits' keyword in 'options' for other limits	1e40
restore_neutron	0|1	Not functional for Union version	0
radius	m	Radius of sphere/banana shape monitor	0
options	str	String that specifies the configuration of the monitor. The general syntax is "[x] options..." (see Descr.).	"NULL"
filename	str	Output file name (overrides file=XX option).	"NULL"
geometry	str	Name of an OFF file to specify a complex geometry detector	"NULL"
nowritefile	1	If set, logger will skip writing to disk	0
nexus_bins	1	NeXus mode only: store component BIN information (-1 disable, 0 enable for list mode monitor, 1 enable for any montor)	0
username0	str	Name assigned to User0	"NULL"
username1	str	Name assigned to User1	"NULL"
username2	str	Name assigned to User2	"NULL"
username3	str	Name assigned to User3	"NULL"
username4	str	Name assigned to User4	"NULL"
username5	str	Name assigned to User5	"NULL"
username6	str	Name assigned to User6	"NULL"
username7	str	Name assigned to User7	"NULL"
username8	str	Name assigned to User8	"NULL"
username9	str	Name assigned to User9	"NULL"

Name	Unit	Description	Default
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_abs_logger_nD.comp`.
- See also the Monitor_nD mcdoc page"

12.46. The `Union_abs_logger_nD_scintillator` McStas Component

Absorption logger for a specific position sensitive scintillator detector

Identification

- **Author:** Milán Klausz, derived from Union_abs_logger_nD by Mads Bertelsen
- **Origin:** ESS
- **Date:** 23.09.26

Description

Part of the Union components, a set of components that work together and thus separates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) Logger and conditional components can be placed which will record what happens 5) A Union_master component placed after all of the above

Only in step 5 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

This component is an absorption logger, and thus placed in point 4) above.

An absorption logger will log something for each absorption event happening in the geometry or geometries on which it is attached. These are specified in the target_geometry string. By leaving it blank, all geometries are logged, even the ones not defined at this point in the instrument file. Multiple geometries are specified as a comma separated list.

This absorption logger stores absorption as events, with position, velocity, time and weight. The Monitor_nD libraries are used to write the event files. This version is a close copy of Monitor_nD, having the same interface, though the user must be aware that no propagation happens for rays to hit the detector pixels, instead it uses the position where the ray was absorbed. Values must still be set for xwidth and yheight: they do not select which absorptions are recorded, that is decided by the Union geometry named in target_geometry, but a monitor with zero area is deactivated by Monitor_nD.

This absorption logger needs to be placed in space, the position and velocity is recorded in the coordinate system of the logger component.

It is possible to attach one or more conditional components to this absorption logger. Such a conditional component would impose a condition on the state of the neutron after the Union_master component that executes the simulation, and the absorption logger will only record the event if this condition is true.

To use the logger_conditional_extend function, set it to some integer value n and make and extend section to the master component that runs the geometry. In this extend function, logger_conditional_extend[n] is 1 if the conditional stack evaluated to true, 0 if not. This way one can check what rays is logged using regular McStas monitors. Only works if a conditional is applied to this logger.

This is a scintillator variant of Union_abs_logger_nD, designed for a specific position sensitive scintillator detector: a GS20 converter layer read out by a multi-anode photomultiplier tube (MAPMT). The logger is placed in the converter layer, and represents the physics from the neutron conversion up to a detection event being triggered in one of the MAPMT pixels.

Instead of logging each absorption at its own position, it converts it into up to five weighted detection events: the MAPMT pixel facing the absorption position and its four neighbours. The weights are tabulated position dependent detection efficiencies, read from the text files true/up/down/left/right_pixel_hit_efficiency[_high_resolution].txt (must be present in the working directory). They answer the question "given that a neutron is absorbed at this position in the converter, what is the probability that a detection event is recorded in each MAPMT pixel", and were produced by a separate Geant4 simulation of the scintillation light, applying a detection threshold per pixel. That threshold suppresses events whose light is shared between neighbouring pixels and would otherwise be counted twice. Two MAPMT pixel sizes are supported, each with its own set of tables.

Note that the detector geometry is hard coded to match the tabulated efficiency files, and is not derived from xwidth and yheight: the tables are read as a 100 x 100 grid of 0.5 mm bins covering a 50 x 50 mm converter tile, read out by a MAPMT pixel grid spanning the central 48.5 x 48.5 mm of that tile. Absorptions outside that map are clamped to its edge bins rather than rejected, so attaching this logger to a geometry of a different size silently gives meaningless efficiencies. Using it for another detector means replacing the efficiency files and adjusting these hard coded values together.

The real MAPMT pixels are not actually all the same size: the outermost ring of pixels is 0.25 mm larger than the rest (3.25 mm vs 3 mm for the high resolution grid, 6.25 mm vs 6 mm for the low resolution one). Monitor_nD only supports a uniform pixel size, so it

cannot represent that directly. The trick used here is to build the Monitor_nD pixel grid at the regular (smaller) pixel size, spanning the central 48 x 48 mm, and to treat the 0.25 mm wide band just outside it as still belonging to the perimeter pixels: an absorption position that falls in that band is shifted inward by 0.25 mm - onto the strip of the grid that Monitor_nD considers part of the corresponding (undersized) perimeter pixel - before the position is handed to Monitor_nD, so that Monitor_nD bins the resulting detection event into the correct perimeter pixel. This repositioning is done purely for pixel assignment, after the pixel-hit detection efficiencies have already been looked up for the true, unshifted absorption position (see compute_scintillator_pixel_hit_probabilities and move_perimeter_det_event_inside_monitor_limits below).

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_geometry	string	Comma separated list of geometry names that will be logged, leave empty for all volumes (even not defined yet)	"NULL"
order_total	1	Only log rays that have scattered n times, -1 for all orders	-1
order_volume	1	Only log rays that have scattered n times in the same geometry, -1 for all orders	-1
logger_conditional_extend_index	1	If a conditional is used with this logger, the result of each conditional calculation can be made available in extend as a array called "logger_conditional_extend", and one would then access logger_conditional_extend[n] if logger_conditional_extend_index is set to n	-1
user0	str	Variable name of USERVAR to be monitored by user0.	""
user1	str	Variable name of USERVAR to be monitored by user1.	""
user2	str	Variable name of USERVAR to be monitored by user2.	""
user3	str	Variable name of USERVAR to be monitored by user3.	""
user4	str	Variable name of USERVAR to be monitored by user4.	""
user5	str	Variable name of USERVAR to be monitored by user5.	""

Name	Unit	Description	Default
user6	str	Variable name of USERVAR to be monitored by user6.	""
user7	str	Variable name of USERVAR to be monitored by user7.	""
user8	str	Variable name of USERVAR to be monitored by user8.	""
user9	str	Variable name of USERVAR to be monitored by user9.	""
xwidth	m	Width of detector.	0
yheight	m	Height of detector.	0
zdepth	m	Thickness of detector (z).	0
xmin	m	Lower x bound of opening	0
xmax	m	Upper x bound of opening	0
ymin	m	Lower y bound of opening	0
ymax	m	Upper y bound of opening	0
zmin	m	Lower z bound of opening	0
zmax	m	Upper z bound of opening	0
bins	1	Number of bins to force for all variables. Use 'bins' keyword in 'options' for heterogeneous bins	0
min	u	Minimum range value to force for all variables. Use 'min' or 'limits' keyword in 'options' for other limits	-1e40
max	u	Maximum range value to force for all variables. Use 'max' or 'limits' keyword in 'options' for other limits	1e40
restore_neutron	0|1	Not functional for Union version	0
radius	m	Radius of sphere/banana shape monitor	0
options	str	String that specifies the configuration of the monitor. The general syntax is "[x] options..." (see Descr.).	"NULL"
filename	str	Output file name (overrides file=XX option).	"NULL"
geometry	str	Name of an OFF file to specify a complex geometry detector	"NULL"
nowritefile	1	If set, logger will skip writing to disk	0
nexus_bins	1	NeXus mode only: store component BIN information (-1 disable, 0 enable for list mode monitor, 1 enable for any montor)	0
username0	str	Name assigned to User0	"NULL"
username1	str	Name assigned to User1	"NULL"

Name	Unit	Description	Default
username2	str	Name assigned to User2	"NULL"
username3	str	Name assigned to User3	"NULL"
username4	str	Name assigned to User4	"NULL"
username5	str	Name assigned to User5	"NULL"
username6	str	Name assigned to User6	"NULL"
username7	str	Name assigned to User7	"NULL"
username8	str	Name assigned to User8	"NULL"
username9	str	Name assigned to User9	"NULL"
is_high_resolution	0|1	Selects the MAPMT pixel size (0: low resolution, 6 mm pixels; 1: high resolution, 3 mm pixels), and correspondingly which set of pixel-hit-efficiency table files is read.	0
table_dir	str	Directory of the table files (default: none)	""
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_abs_logger_nD_scintillator.comp`.
- See also the Monitor_nD mcdoc page"

12.47. Union conditionals

12.48. The `Union_conditional_PSD` McStas Component

A conditional component in the shape of an area the neutrons must hit

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box/cylinder/sphere, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union components

This is a conditional component that affects the loggers in the target_loggers string. When a logger is affected, it will only record events if the conditional is true at the end of the simulation of a ray in the master. Conditionals can be used to for example limit the loggers to rays that end within a certain energy range, time interval or similar.

One can apply several conditionals to each logger if desired.

In the extend section of a master, the tagging conditional can be acsessed by the variable name tagging_conditional_extend. Beware, that it only works as long as the tagging system is active, so you may want to increase the number of histories allowed by that master component before stopping.

This conditional is a little special, because it needs to be placed in space. It's center location is like a psd.

overwrite_logger_weight can be used to force the loggers this conditional controls to write the final weight for each scattering event, instead of the recorded value.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_loggers	string	Comma seperated list of loggers this conditional should affect	"NULL"
master_tagging	1	Apply this conditional to the tagging system in next master	0
tagging_extend_index	1	Not currently used	-1
xwidth	m	Width of target area	
yheight	m	Height of target area	
time_min	s	Min time (on psd), if not set ignored	0
time_max	s	Max time (on psd), if not set ignored	0
overwrite_logger_weight	1	Default = 0, overwrite = 1	0

Name	Unit	Description	Default
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_conditional_PSD.comp`.

12.49. The `Union_conditional_standard` McStas Component

A conditional component that filter on basic variables of exit state

Identification

- **Author:** Mads Bertelsen
- **Origin:** University of Copenhagen
- **Date:** 20.08.15

Description

Part of the Union components, a set of components that work together and thus sperates geometry and physics within McStas. The use of this component requires other components to be used.

1) One specifies a number of processes using process components 2) These are gathered into material definitions using Union_make_material 3) Geometries are placed using Union_box / Union_cylinder, assigned a material 4) A Union_master component placed after all of the above

Only in step 4 will any simulation happen, and per default all geometries defined before this master, but after the previous will be simulated here.

There is a dedicated manual available for the Union_components

This is a conditional component that affects the loggers in the target_loggers string. When a logger is affected, it will only record events if the conditional is true at the end of the simulation of a ray in the master. Conditionals can be used to for example limit the loggers to rays that end within a certain energy range, time interval or similar.

One can apply several conditionals to each logger if desired.

In the extend section of a master, the tagging conditional can be acsessed by the variable name tagging_conditional_extend. Beware, that it only works as long as the tagging system is active, so you may want to increase the number of histories allowed by that master component before stopping.

overwrite_logger_weight can be used to force the loggers this conditional controls to write the final weight for each scattering event, instead of the recorded value.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_loggers	string	Comma seperated list of loggers this conditional should affect	"NULL"
master_tagging	1	Apply this conditional to the tagging system in next master	0
tagging_extend_index	1	Not currently used	-1
time_min	s	Min time, if not set ignored	0
time_max	s	Max time, if not set ignored	0
E_min	meV	Max energy, if not set ignored	0
E_max	meV	Min energy, if not set ignored	0
total_order_min	1	Min scattering order, if not set ignored	0
total_order_max	1	Max scattering order, if not set ignored	0
last_volume_index	1	Not used yet	-1
overwrite_logger_weight	1	Default = 0, overwrite = 1	0
init	string	Deprecated and unused. Accepted so that instruments written for McStas/McXtrace 3.8.7 and earlier, which name the Union_init component here, still compile.	""

Links

- Component source code found in file `Union_conditional_standard.comp`.

13. SasView/sasmodels form factors: small-angle scattering

This chapter documents the `sasmodels` category: a large set of small-angle scattering (SANS/USANS) form-factor models, generated automatically from the open-source SasView/sasmodels project (`https://github.com/SasView/sasmodels`) and exposed to McStas as individual `SasView_*` components (each wrapping one sasmodels kernel; they supersede the older, generic `SasView_model` component, which is now in the `obsolete` category). This gives access to the same, actively maintained library of scattering-law implementations used for SANS data fitting in SasView itself, directly as McStas sample components, without needing to hand-implement each form factor.

Most models exist in a plain (orientation-averaged, i.e. powder/isotropic) form and, where the shape has a well-defined orientation, an `_aniso` variant that additionally accepts orientation parameters. For the underlying physics of any individual model (defining equation, parameter meaning, validity range), consult the corresponding page in the SasView documentation (`https://www.sasview.org/docs/`); the parameter tables below are generated directly from each component's McDoc header and give the McStas-specific parameter names, units and defaults.

See also `Templates/templateSasView` and `Templates/Test_SasView_guinier` in the example instrument library for worked examples, and the `samples/SANS_*` and `contrib/SANS*` components (documented in the Samples chapter) for McStas's own, native (non-SasView-derived) SANS sample models.

13.1. SasView form-factor components

13.2. The `SasView_adsorbed_layer` McStas Component

SasView adsorbed_layer model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_adsorbed_layer component, generated from adsorbed_layer.c in sasmodels.

Example: SasView_adsorbed_layer(second_moment, adsorbed_amount, density_shell, radius, volfraction, sld_shell, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
second_moment	Å	([0.0, inf]) Second moment of polymer distribution.	23.0
adsorbed_amount	mg/m^2	([0.0, inf]) Adsorbed amount of polymer.	1.9
density_shell	g/cm^3	([0.0, inf]) Bulk density of polymer in the shell.	0.7
radius	Å	([0.0, inf]) Core particle radius.	500.0
volfraction	None	([0.0, inf]) Core particle volume fraction.	0.14
sld_shell	1e-6/Å^2	([-inf, inf]) Polymer shell SLD.	1.5
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent SLD. Optional parameters:	6.3
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5

Name	Unit	Description	Default
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_adsorbed_layer.comp`.

13.3. The `SasView_barbell` McStas Component

SasView barbell model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_barbell component, generated from barbell.c in sasmodels.

Example: SasView_barbell(sld, sld_solvent, radius_bell, radius, length, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_bell=0.0, pd_radius=0.0, pd_length=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Barbell scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1
radius_bell	Å	([0, inf]) Spherical bell radius.	40

Name	Unit	Description	Default
radius	Å	([0, inf]) Cylindrical bar radius.	20
length	Å	([0, inf]) Cylinder bar length. Optional parameters:	400
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius_bell		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_barbell.comp`.

13.4. The `SasView_barbell_aniso` McStas Component

SasView barbell model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_barbell component, generated from barbell.c in sasmodels.
Example: SasView_barbell_aniso(sld, sld_solvent, radius_bell, radius, length, theta, phi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_bell=0.0, pd_radius=0.0, pd_length=0.0, pd_theta=0.0, pd_phi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Barbell scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1
radius_bell	Å	([0, inf]) Spherical bell radius.	40
radius	Å	([0, inf]) Cylindrical bar radius.	20
length	Å	([0, inf]) Cylinder bar length. Optional parameters:	400
theta			60
phi			60
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01

Name	Unit	Description	Default
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius_bell		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_barbell_aniso.comp`.

13.5. The `SasView_bcc_paracrystal` McStas Component

SasView bcc_paracrystal model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_bcc_paracrystal component, generated from bcc_paracrystal.c in sasmodels.

Example: SasView_bcc_paracrystal(dnn, d_factor, radius, sld, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
dnn	Å	([-inf, inf]) Nearest neighbour distance.	220
d_factor		([-inf, inf]) Paracrystal distortion factor.	0.06
radius	Å	([0, inf]) Particle radius.	40
sld	1e-6/Å^2	([-inf, inf]) Particle scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	1
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1

Name	Unit	Description	Default
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_bcc_paracrystal.comp`.

13.6. The `SasView_bcc_paracrystal_aniso` McStas Component

SasView bcc_paracrystal model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_bcc_paracrystal component, generated from bcc_paracrystal.c in sasmodels.
Example: SasView_bcc_paracrystal_aniso(dnn, d_factor, radius, sld, sld_solvent, theta, phi, Psi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_theta=0.0, pd_phi=0.0, pd_Psi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
dnn	Å	([-inf, inf]) Nearest neighbour distance.	220
d_factor		([-inf, inf]) Paracrystal distortion factor.	0.06
radius	Å	([0, inf]) Particle radius.	40
sld	1e-6/Å^2	([-inf, inf]) Particle scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	1
theta			60
phi			60
Psi			60
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0

Name	Unit	Description	Default
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_Psi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_bcc_paracrystal_aniso.comp`.

13.7. The `SasView_binary_hard_sphere` McStas Component

SasView binary_hard_sphere model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_binary_hard_sphere component, generated from binary_hard_sphere.c in sasmodels.

Example: SasView_binary_hard_sphere(radius_lg, radius_sm, volfraction_lg, volfraction_sm, sld_lg, sld_sm, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_lg=0.0, pd_radius_sm=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius_lg	Å	([0, inf]) radius of large particle.	100
radius_sm	Å	([0, inf]) radius of small particle.	25
volfraction_lg		([0, 1]) volume fraction of large particle.	0.1
volfraction_sm		([0, 1]) volume fraction of small particle.	0.2

Name	Unit	Description	Default
sld_lg	1e-6/Å^2	([-inf, inf]) scattering length density of large particle.	3.5
sld_sm	1e-6/Å^2	([-inf, inf]) scattering length density of small particle.	0.5
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	6.36
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius_lg		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_radius_sm		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_binary_hard_sphere.comp`.

13.8. The `SasView_broad_peak` McStas Component

SasView broad_peak model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_broad_peak component, generated from broad_peak.c in sasmodels.
Example: SasView_broad_peak(porod_scale, porod_exp, peak_scale, correlation_length, peak_pos, width_exp, shape_exp, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_correlation_leng

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
porod_scale		([-inf, inf]) Power law scale factor.	1e-05
porod_exp		([-inf, inf]) Exponent of power law.	3.0
peak_scale		([-inf, inf]) Scale factor for broad peak.	10.0
correlation_length	Å	([-inf, inf]) screening length.	50.0
peak_pos	1/Å	([-inf, inf]) Peak position in q.	0.1
width_exp		([-inf, inf]) Exponent of peak width.	2.0
shape_exp		([-inf, inf]) Exponent of peak shape. Optional parameters:	1.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005

Name	Unit	Description	Default
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_correlation_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_broad_peak.comp`.

13.9. The `SasView_capped_cylinder` McStas Component

SasView capped_cylinder model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_capped_cylinder component, generated from capped_cylinder.c in sasmodels.
Example: SasView_capped_cylinder(sld, sld_solvent, radius, radius_cap, length, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_radius_cap=0.0, pd_length=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Cylinder scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1
radius	Å	([0, inf]) Cylinder radius.	20
radius_cap	Å	([0, inf]) Cap radius.	20
length	Å	([0, inf]) Cylinder length. Optional parameters:	400
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0

Name	Unit	Description	Default
pd_radius_cap		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_capped_cylinder.comp`.

13.10. The `SasView_capped_cylinder_aniso` McStas Component

SasView capped_cylinder model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_capped_cylinder component, generated from capped_cylinder.c in sasmodels.

Example: SasView_capped_cylinder_aniso(sld, sld_solvent, radius, radius_cap, length, theta, phi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_radius_cap=0.0, pd_length=0.0, pd_theta=0.0, pd_phi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Cylinder scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1

Name	Unit	Description	Default
radius	Å	([0, inf]) Cylinder radius.	20
radius_cap	Å	([0, inf]) Cap radius.	20
length	Å	([0, inf]) Cylinder length. Optional parameters:	400
theta			60
phi			60
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_radius_cap		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0

Name	Unit	Description	Default
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_capped_cylinder_aniso.comp`.

13.11. The `SasView_core_multi_shell` McStas Component

SasView core_multi_shell model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_core_multi_shell component, generated from core_multi_shell.c in sasmodels.

Example: SasView_core_multi_shell(sld_core, radius, sld_solvent, n, sld[n], thickness[n], model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_thickness[n]=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default

Links

- Component source code found in file `SasView_core_multi_shell.comp`.

13.12. The `SasView_core_shell_bicelle` McStas Component

SasView core_shell_bicelle model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_core_shell_bicelle component, generated from core_shell_bicelle.c in sasmodels.

Example: SasView_core_shell_bicelle(radius, thick_rim, thick_face, length, sld_core, sld_face, sld_rim, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_thick_rim=
pd_thick_face=0.0, pd_length=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	Å	([0, inf]) Cylinder core radius.	80
thick_rim	Å	([0, inf]) Rim shell thickness.	10
thick_face	Å	([0, inf]) Cylinder face thickness.	10
length	Å	([0, inf]) Cylinder length.	50
sld_core	1e-6/Å^2	([-inf, inf]) Cylinder core scattering length density.	1
sld_face	1e-6/Å^2	([-inf, inf]) Cylinder face scattering length density.	4
sld_rim	1e-6/Å^2	([-inf, inf]) Cylinder rim scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	1
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0

Name	Unit	Description	Default
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_rim		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_face		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_core_shell_bicelle.comp`.

13.13. The `SasView_core_shell_bicelle_aniso` McStas Component

SasView core_shell_bicelle model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_core_shell_bicelle component, generated from core_shell_bicelle.c in sasmodels.

Example: SasView_core_shell_bicelle_aniso(radius, thick_rim, thick_face, length, sld_core, sld_face, sld_rim, sld_solvent, theta, phi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_thick_rim=0.0, pd_thick_face=0.0, pd_length=0.0, pd_theta=0.0, pd_phi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	Å	([0, inf]) Cylinder core radius.	80
thick_rim	Å	([0, inf]) Rim shell thickness.	10
thick_face	Å	([0, inf]) Cylinder face thickness.	10
length	Å	([0, inf]) Cylinder length.	50
sld_core	1e-6/Å^2	([-inf, inf]) Cylinder core scattering length density.	1
sld_face	1e-6/Å^2	([-inf, inf]) Cylinder face scattering length density.	4
sld_rim	1e-6/Å^2	([-inf, inf]) Cylinder rim scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	1
theta			90
phi			0

Name	Unit	Description	Default
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_rim		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_face		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0

Name	Unit	Description	Default
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_core_shell_bicelle_aniso.comp`.

13.14. The `SasView_core_shell_bicelle_elliptical` McStas Component

SasView core_shell_bicelle_elliptical model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_core_shell_bicelle_elliptical component, generated from core_shell_bicelle_elliptical.c in sasmodels.

Example: SasView_core_shell_bicelle_elliptical(radius, x_core, thick_rim, thick_face, length, sld_core, sld_face, sld_rim, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_thick_rim=0.0, pd_thick_face=0.0, pd_length=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	Å	([0, inf]) Cylinder core radius r_minor.	30
x_core	None	([0, inf]) Axial ratio of core, X = r_major/r_minor.	3
thick_rim	Å	([0, inf]) Rim shell thickness.	8
thick_face	Å	([0, inf]) Cylinder face thickness.	14
length	Å	([0, inf]) Cylinder length.	50

Name	Unit	Description	Default
sld_core	1e-6/Å^2	([-inf, inf]) Cylinder core scattering length density.	4
sld_face	1e-6/Å^2	([-inf, inf]) Cylinder face scattering length density.	7
sld_rim	1e-6/Å^2	([-inf, inf]) Cylinder rim scattering length density.	1
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	6
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_rim		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0

Name	Unit	Description	Default
pd_thick_face		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_core_shell_bicelle_elliptical.comp`.

13.15. The `SasView_core_shell_bicelle_elliptical_aniso` McStas Component

SasView core_shell_bicelle_elliptical model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_core_shell_bicelle_elliptical component, generated from core_shell_bicelle_elliptical.c in sasmodels.

Example: SasView_core_shell_bicelle_elliptical_aniso(radius, x_core, thick_rim, thick_face, length, sld_core, sld_face, sld_rim, sld_solvent, theta, phi, Psi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_thick_rim=0.0, pd_thick_face=0.0, pd_length=0.0, pd_theta=0.0, pd_phi=0.0, pd_Psi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	Å	([0, inf]) Cylinder core radius r_minor.	30

Name	Unit	Description	Default
x_core	None	([0, inf]) Axial ratio of core, X = r_major/r_minor.	3
thick_rim	Å	([0, inf]) Rim shell thickness.	8
thick_face	Å	([0, inf]) Cylinder face thickness.	14
length	Å	([0, inf]) Cylinder length.	50
sld_core	1e-6/Å^2	([-inf, inf]) Cylinder core scattering length density.	4
sld_face	1e-6/Å^2	([-inf, inf]) Cylinder face scattering length density.	7
sld_rim	1e-6/Å^2	([-inf, inf]) Cylinder rim scattering length density.	1
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	6
theta			90.0
phi			0
Psi			0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0

Name	Unit	Description	Default
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_rim		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_face		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_Psi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_core_shell_bicelle_elliptical_aniso.comp`.

13.16. The `SasView_core_shell_bicelle_elliptical_belt_rough` McStas Component

SasView core_shell_bicelle_elliptical_belt_rough model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_core_shell_bicelle_elliptical_belt_rough component, generated from core_shell_bicelle_elliptical_bel
in sasmodels.

Example: SasView_core_shell_bicelle_elliptical_belt_rough(radius, x_core, thick_rim, thick_face, length, sld_core, sld_face, sld_rim, sld_solvent, sigma, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_thick_rim=0.0, pd_thick_face=0.0, pd_length=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	Å	([0, inf]) Cylinder core radius r_minor.	30
x_core	None	([0, inf]) Axial ratio of core, X = r_major/r_minor.	3
thick_rim	Å	([0, inf]) Rim or belt shell thickness.	8
thick_face	Å	([0, inf]) Cylinder face thickness.	14
length	Å	([0, inf]) Cylinder length.	50
sld_core	1e-6/Å^2	([-inf, inf]) Cylinder core scattering length density.	4
sld_face	1e-6/Å^2	([-inf, inf]) Cylinder face scattering length density.	7
sld_rim	1e-6/Å^2	([-inf, inf]) Cylinder rim scattering length density.	1
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	6
sigma	Å	([0, inf]) Interfacial roughness. Optional parameters:	0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005

Name	Unit	Description	Default
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_rim		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_face		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_core_shell_bicelle_elliptical_belt_rough.c`

13.17. The `SasView_core_shell_bicelle_elliptical_belt_rough_aniso` McStas Component

SasView core_shell_bicelle_elliptical_belt_rough model component as sample description.

Identification

- **Author:** Jose Robledo

- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_core_shell_bicelle_elliptical_belt_rough component, generated from core_shell_bicelle_elliptical_bel
in sasmodels.

Example: SasView_core_shell_bicelle_elliptical_belt_rough_aniso(radius, x_core, thick_rim, thick_face, length, sld_core, sld_face, sld_rim, sld_solvent, sigma, theta, phi, Psi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_thick_rim=0.0, pd_thick_face=0.0, pd_length=0.0, pd_theta=0.0, pd_phi=0.0, pd_Psi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	Å	([0, inf]) Cylinder core radius r_minor.	30
x_core	None	([0, inf]) Axial ratio of core, X = r_major/r_minor.	3
thick_rim	Å	([0, inf]) Rim or belt shell thickness.	8
thick_face	Å	([0, inf]) Cylinder face thickness.	14
length	Å	([0, inf]) Cylinder length.	50
sld_core	1e-6/Å^2	([-inf, inf]) Cylinder core scattering length density.	4
sld_face	1e-6/Å^2	([-inf, inf]) Cylinder face scattering length density.	7
sld_rim	1e-6/Å^2	([-inf, inf]) Cylinder rim scattering length density.	1
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	6
sigma	Å	([0, inf]) Interfacial roughness. Optional parameters:	0
theta			90.0
phi			0
Psi			0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0

Name	Unit	Description	Default
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_rim		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_face		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0

Name	Unit	Description	Default
pd_Psi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_core_shell_bicelle_elliptical_belt_rough_aniso.comp`.

13.18. The `SasView_core_shell_cylinder` McStas Component

SasView core_shell_cylinder model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_core_shell_cylinder component, generated from core_shell_cylinder.c in sasmodels.

Example: SasView_core_shell_cylinder(sld_core, sld_shell, sld_solvent, radius, thickness, length, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_thickness=0.0, pd_length=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld_core	1e-6/Å^2	([-inf, inf]) Cylinder core scattering length density.	4
sld_shell	1e-6/Å^2	([-inf, inf]) Cylinder shell scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1

Name	Unit	Description	Default
radius	Å	([0, inf]) Cylinder core radius.	20
thickness	Å	([0, inf]) Cylinder shell thickness.	20
length	Å	([0, inf]) Cylinder length. Optional parameters:	400
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thickness		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_core_shell_cylinder.comp`.

13.19. The `SasView_core_shell_cylinder_aniso` McStas Component

SasView core_shell_cylinder model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_core_shell_cylinder component, generated from core_shell_cylinder.c in sasmodels.

Example: SasView_core_shell_cylinder_aniso(sld_core, sld_shell, sld_solvent, radius, thickness, length, theta, phi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_thickness=0.0, pd_length=0.0, pd_theta=0.0, pd_phi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld_core	1e-6/Å^2	([-inf, inf]) Cylinder core scattering length density.	4
sld_shell	1e-6/Å^2	([-inf, inf]) Cylinder shell scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1
radius	Å	([0, inf]) Cylinder core radius.	20
thickness	Å	([0, inf]) Cylinder shell thickness.	20
length	Å	([0, inf]) Cylinder length. Optional parameters:	400
theta			60
phi			60

Name	Unit	Description	Default
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thickness		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_core_shell_cylinder_aniso.comp`.

13.20. The `SasView_core_shell_ellipsoid` McStas Component

SasView core_shell_ellipsoid model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_core_shell_ellipsoid component, generated from core_shell_ellipsoid.c in sasmodels.

Example: SasView_core_shell_ellipsoid(radius_equat_core, x_core, thick_shell, x_polar_shell, sld_core, sld_shell, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_equat_core=0.0, pd_thick_shell=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius_equat_core	Å	([0, inf]) Equatorial radius of core.	20
x_core	None	([0, inf]) axial ratio of core, X = r_polar/r_equatorial.	3
thick_shell	Å	([0, inf]) thickness of shell at equator.	30
x_polar_shell		([0, inf]) ratio of thickness of shell at pole to that at equator.	1
sld_core	1e-6/Å^2	([-inf, inf]) Core scattering length density.	2
sld_shell	1e-6/Å^2	([-inf, inf]) Shell scattering length density.	1
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	6.3

Name	Unit	Description	Default
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius_equat_core		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_shell		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_core_shell_ellipsoid.comp`.

13.21. The `SasView_core_shell_ellipsoid_aniso` McStas Component

SasView core_shell_ellipsoid model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_core_shell_ellipsoid component, generated from core_shell_ellipsoid.c in sasmodels.

Example: SasView_core_shell_ellipsoid_aniso(radius_equat_core, x_core, thick_shell, x_polar_shell, sld_core, sld_shell, sld_solvent, theta, phi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_equat_core=0.0, pd_thick_shell=0.0, pd_theta=0.0, pd_phi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius_equat_core	Å	([0, inf]) Equatorial radius of core.	20
x_core	None	([0, inf]) axial ratio of core, X = r_polar/r_equatorial.	3
thick_shell	Å	([0, inf]) thickness of shell at equator.	30
x_polar_shell		([0, inf]) ratio of thickness of shell at pole to that at equator.	1
sld_core	1e-6/Å^2	([-inf, inf]) Core scattering length density.	2
sld_shell	1e-6/Å^2	([-inf, inf]) Shell scattering length density.	1
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	6.3
theta			0
phi			0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0

Name	Unit	Description	Default
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius_equat_core		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_shell		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_core_shell_ellipsoid_aniso.comp`.

13.22. The `SasView_core_shell_parallelepiped` McStas Component

SasView core_shell_parallelepiped model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_core_shell_parallelepiped component, generated from core_shell_parallelepiped.c in sasmodels.

Example: SasView_core_shell_parallelepiped(sld_core, sld_a, sld_b, sld_c, sld_solvent, length_a, length_b, length_c, thick_rim_a, thick_rim_b, thick_rim_c, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_length_a=0.0, pd_length_b=0.0, pd_length_c=0.0, pd_thick_rim_a=0.0, pd_thick_rim_b=0.0, pd_thick_rim_c=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld_core	1e-6/Å^2	([-inf, inf]) Parallelepiped core scattering length density.	1
sld_a	1e-6/Å^2	([-inf, inf]) Parallelepiped A rim scattering length density.	2
sld_b	1e-6/Å^2	([-inf, inf]) Parallelepiped B rim scattering length density.	4
sld_c	1e-6/Å^2	([-inf, inf]) Parallelepiped C rim scattering length density.	2
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	6
length_a	Å	([0, inf]) Shorter side of the parallelepiped.	35
length_b	Å	([0, inf]) Second side of the parallelepiped.	75
length_c	Å	([0, inf]) Larger side of the parallelepiped.	400
thick_rim_a	Å	([0, inf]) Thickness of A rim.	10
thick_rim_b	Å	([0, inf]) Thickness of B rim.	10
thick_rim_c	Å	([0, inf]) Thickness of C rim. Optional parameters:	10

Name	Unit	Description	Default
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_length_a		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length_b		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length_c		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_rim_a		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_rim_b		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0

Name	Unit	Description	Default
pd_thick_rim_c		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_core_shell_parallelepiped.comp`.

13.23. The `SasView_core_shell_parallelepiped_aniso` McStas Component

SasView core_shell_parallelepiped model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_core_shell_parallelepiped component, generated from core_shell_parallelepiped.c in sasmodels.

Example: SasView_core_shell_parallelepiped_aniso(sld_core, sld_a, sld_b, sld_c, sld_solvent, length_a, length_b, length_c, thick_rim_a, thick_rim_b, thick_rim_c, theta, phi, Psi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_length_a=0.0, pd_length_b=0.0, pd_length_c=0.0, pd_thick_rim_a=0.0, pd_thick_rim_b=0.0, pd_thick_rim_c=0.0, pd_theta=0.0, pd_phi=0.0, pd_Psi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld_core	1e-6/Å^2	([-inf, inf]) Parallelepiped core scattering length density.	1
sld_a	1e-6/Å^2	([-inf, inf]) Parallelepiped A rim scattering length density.	2

Name	Unit	Description	Default
sld_b	1e-6/Å^2	([-inf, inf]) Parallelepiped B rim scattering length density.	4
sld_c	1e-6/Å^2	([-inf, inf]) Parallelepiped C rim scattering length density.	2
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	6
length_a	Å	([0, inf]) Shorter side of the parallelepiped.	35
length_b	Å	([0, inf]) Second side of the parallelepiped.	75
length_c	Å	([0, inf]) Larger side of the parallelepiped.	400
thick_rim_a	Å	([0, inf]) Thickness of A rim.	10
thick_rim_b	Å	([0, inf]) Thickness of B rim.	10
thick_rim_c	Å	([0, inf]) Thickness of C rim. Optional parameters:	10
theta			0
phi			0
Psi			0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0

Name	Unit	Description	Default
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_length_a		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length_b		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length_c		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_rim_a		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_rim_b		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_rim_c		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_Psi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_core_shell_parallelepiped_aniso.comp`.

13.24. The `SasView_core_shell_sphere` McStas Component

SasView core_shell_sphere model component as sample description.

Identification

- **Author:** Jose Robledo

- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_core_shell_sphere component, generated from core_shell_sphere.c in sasmodels.

Example: SasView_core_shell_sphere(radius, thickness, sld_core, sld_shell, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_thickness=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	Å	([0, inf]) Sphere core radius.	60.0
thickness	Å	([0, inf]) Sphere shell thickness.	10.0
sld_core	1e-6/Å^2	([-inf, inf]) core scattering length density.	1.0
sld_shell	1e-6/Å^2	([-inf, inf]) shell scattering length density.	2.0
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	3.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1

Name	Unit	Description	Default
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thickness		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_core_shell_sphere.comp`.

13.25. The `SasView_correlation_length` McStas Component

SasView correlation_length model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_correlation_length component, generated from correlation_length.c in sasmodels.

Example: SasView_correlation_length(lorentz_scale, porod_scale, cor_length, porod_exp, lorentz_exp, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_cor_length=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
lorentz_scale		([0, inf]) Lorentzian Scaling Factor.	10.0
porod_scale		([0, inf]) Porod Scaling Factor.	1e-06
cor_length	Å	([0, inf]) Correlation length, xi, in Lorentzian.	50.0
porod_exp		([0, inf]) Porod Exponent, n, in q^-n.	3.0
lorentz_exp		([0, inf]) Lorentzian Exponent, m, in 1/(1 + (q.xi)^m). Optional parameters:	2.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_cor_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_correlation_length.comp`.

13.26. The `SasView_cylinder` McStas Component

SasView cylinder model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_cylinder component, generated from cylinder.c in sasmodels.
Example: SasView_cylinder(sld, sld_solvent, radius, length, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_length=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Cylinder scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1
radius	Å	([0, inf]) Cylinder radius.	20
length	Å	([0, inf]) Cylinder length. Optional parameters:	400
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005

Name	Unit	Description	Default
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_cylinder.comp`.

13.27. The `SasView_cylinder_aniso` McStas Component

SasView cylinder model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_cylinder component, generated from cylinder.c in sasmodels.

Example: SasView_cylinder_aniso(sld, sld_solvent, radius, length, theta, phi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1,

target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_length=0.0, pd_theta=0.0, pd_phi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Cylinder scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1
radius	Å	([0, inf]) Cylinder radius.	20
length	Å	([0, inf]) Cylinder length. Optional parameters:	400
theta			60
phi			60
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0

Name	Unit	Description	Default
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_cylinder_aniso.comp`.

13.28. The `SasView_dab` McStas Component

SasView dab model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_dab component, generated from dab.c in sasmodels.

Example: SasView_dab(cor_length, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_cor_length=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
cor_length	Å	([0, inf]) correlation length. Optional parameters:	50.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_cor_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_dab.comp`.

13.29. The `SasView_ellipsoid` McStas Component

SasView ellipsoid model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_ellipsoid component, generated from ellipsoid.c in sasmodels.

Example: SasView_ellipsoid(sld, sld_solvent, radius_polar, radius_equatorial, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_polar=0.0, pd_radius_equatorial=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Ellipsoid scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1
radius_polar	Å	([0, inf]) Polar radius.	20
radius_equatorial	Å	([0, inf]) Equatorial radius. Optional parameters:	400
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1

Name	Unit	Description	Default
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius_polar		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_radius_equatorial		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_ellipsoid.comp`.

13.30. The `SasView_ellipsoid_aniso` McStas Component

SasView ellipsoid model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_ellipsoid component, generated from ellipsoid.c in sasmodels.

Example: SasView_ellipsoid_aniso(sld, sld_solvent, radius_polar, radius_equatorial, theta, phi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_polar=0.0, pd_radius_equatorial=0.0, pd_theta=0.0, pd_phi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Ellipsoid scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1
radius_polar	Å	([0, inf]) Polar radius.	20
radius_equatorial	Å	([0, inf]) Equatorial radius. Optional parameters:	400
theta			60
phi			60
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius_polar		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0

Name	Unit	Description	Default
pd_radius_equatorial		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_ellipsoid_aniso.comp`.

13.31. The `SasView_elliptical_cylinder` McStas Component

SasView elliptical_cylinder model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_elliptical_cylinder component, generated from elliptical_cylinder.c in sasmodels.

Example: SasView_elliptical_cylinder(radius_minor, r_ratio, length, sld, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_minor=0.0, pd_length=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius_minor	Å	([0, inf]) Ellipse minor radius.	20.0

Name	Unit	Description	Default
r_ratio		([1, inf]) Ratio of major radius over minor radius.	1.5
length	Å	([1, inf]) Length of the cylinder.	400.0
sld	1e-6/Å^2	([-inf, inf]) Cylinder scattering length density.	4.0
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	1.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius_minor		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_elliptical_cylinder.comp`.

13.32. The `SasView_elliptical_cylinder_aniso` McStas Component

SasView elliptical_cylinder model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_elliptical_cylinder component, generated from elliptical_cylinder.c in sasmodels.

Example: SasView_elliptical_cylinder_aniso(radius_minor, r_ratio, length, sld, sld_solvent, theta, phi, Psi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_minor=0.0, pd_length=0.0, pd_theta=0.0, pd_phi=0.0, pd_Psi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius_minor	Å	([0, inf]) Ellipse minor radius.	20.0
r_ratio		([1, inf]) Ratio of major radius over minor radius.	1.5
length	Å	([1, inf]) Length of the cylinder.	400.0
sld	1e-6/Å^2	([-inf, inf]) Cylinder scattering length density.	4.0
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	1.0
theta			90.0
phi			0
Psi			0

Name	Unit	Description	Default
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius_minor		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_Psi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_elliptical_cylinder_aniso.comp`.

13.33. The `SasView_fcc_paracrystal` McStas Component

SasView fcc_paracrystal model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_fcc_paracrystal component, generated from fcc_paracrystal.c in sasmodels.
Example: SasView_fcc_paracrystal(dnn, d_factor, radius, sld, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
dnn	Å	([-inf, inf]) Nearest neighbour distance.	220
d_factor		([-inf, inf]) Paracrystal distortion factor.	0.06
radius	Å	([0, inf]) Particle radius.	40
sld	1e-6/Å^2	([-inf, inf]) Particle scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	1
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01

Name	Unit	Description	Default
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_fcc_paracrystal.comp`.

13.34. The `SasView_fcc_paracrystal_aniso` McStas Component

SasView fcc_paracrystal model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_fcc_paracrystal component, generated from fcc_paracrystal.c in sasmodels.
Example: SasView_fcc_paracrystal_aniso(dnn, d_factor, radius, sld, sld_solvent, theta, phi, Psi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005,

R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_theta=0.0, pd_phi=0.0, pd_Psi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
dnn	Å	([-inf, inf]) Nearest neighbour distance.	220
d_factor		([-inf, inf]) Paracrystal distortion factor.	0.06
radius	Å	([0, inf]) Particle radius.	40
sld	1e-6/Å^2	([-inf, inf]) Particle scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	1
theta			60
phi			60
Psi			60
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0

Name	Unit	Description	Default
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_Psi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_fcc_paracrystal_aniso.comp`.

13.35. The `SasView_flexible_cylinder` McStas Component

SasView flexible_cylinder model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_flexible_cylinder component, generated from flexible_cylinder.c in sasmodels.
Example: SasView_flexible_cylinder(length, kuhn_length, radius, sld, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_length=0.0, pd_kuhn_length=0.0, pd_radius=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
length	Å	([0, inf]) Length of the flexible cylinder.	1000.0
kuhn_length	Å	([0, inf]) Kuhn length of the flexible cylinder.	100.0
radius	Å	([0, inf]) Radius of the flexible cylinder.	20.0
sld	1e-6/Å^2	([-inf, inf]) Cylinder scattering length density.	1.0
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	6.3
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_kuhn_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0

Name	Unit	Description	Default
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_flexible_cylinder.comp`.

13.36. The `SasView_flexible_cylinder_elliptical` McStas Component

SasView flexible_cylinder_elliptical model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_flexible_cylinder_elliptical component, generated from flexible_cylinder_elliptical.c in sasmodels.

Example: SasView_flexible_cylinder_elliptical(length, kuhn_length, radius, axis_ratio, sld, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_length=0.0, pd_kuhn_length=0.0, pd_radius=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
length	Å	([0, inf]) Length of the flexible cylinder.	1000.0
kuhn_length	Å	([0, inf]) Kuhn length of the flexible cylinder.	100.0
radius	Å	([1, inf]) Radius of the flexible cylinder.	20.0
axis_ratio		([0, inf]) Axis_ratio (major_radius/minor_radius.	1.5

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Cylinder scattering length density.	1.0
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	6.3
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_kuhn_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_flexible_cylinder_elliptical.comp`.

13.37. The `SasView_fractal` McStas Component

SasView fractal model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_fractal component, generated from fractal.c in sasmodels.
Example: SasView_fractal(volfraction, radius, fractal_dim, cor_length, sld_block, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_cor_length=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
volfraction		([0.0, 1]) volume fraction of blocks.	0.05
radius	Å	([0.0, inf]) radius of particles.	5.0
fractal_dim		([0.0, 6.0]) fractal dimension.	2.0
cor_length	Å	([0.0, inf]) cluster correlation length.	100.0
sld_block	1e-6/Å^2	([-inf, inf]) scattering length density of particles.	2.0
sld_solvent	1e-6/Å^2	([-inf, inf]) scattering length density of solvent. Optional parameters:	6.4
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0

Name	Unit	Description	Default
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_cor_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_fractal.comp`.

13.38. The `SasView_fractal_core_shell` McStas Component

SasView fractal_core_shell model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_fractal_core_shell component, generated from fractal_core_shell.c in sasmodels.

Example: SasView_fractal_core_shell(radius, thickness, sld_core, sld_shell, sld_solvent, volfraction, fractal_dim, cor_length, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_thickness=0.0, pd_cor_length=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	Å	([0.0, inf]) Sphere core radius.	60.0
thickness	Å	([0.0, inf]) Sphere shell thickness.	10.0
sld_core	1e-6/Å^2	([-inf, inf]) Sphere core scattering length density.	1.0
sld_shell	1e-6/Å^2	([-inf, inf]) Sphere shell scattering length density.	2.0
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	3.0
volfraction		([0.0, inf]) Volume fraction of building block spheres.	0.05
fractal_dim		([0.0, 6.0]) Fractal dimension.	2.0
cor_length	Å	([0.0, inf]) Correlation length of fractal-like aggregates. Optional parameters:	100.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0

Name	Unit	Description	Default
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thickness		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_cor_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_fractal_core_shell.comp`.

13.39. The `SasView_fuzzy_sphere` McStas Component

SasView fuzzy_sphere model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_fuzzy_sphere component, generated from fuzzy_sphere.c in sasmodels.

Example: SasView_fuzzy_sphere(sld, sld_solvent, radius, fuzziness, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Particle scattering length density.	1
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	3
radius	Å	([0, inf]) Sphere radius.	60
fuzziness	Å	([0, inf]) std deviation of Gaussian convolution for interface (must be << radius). Optional parameters:	10
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_fuzzy_sphere.comp`.

13.40. The `SasView_gauss_lorentz_gel` McStas Component

SasView gauss_lorentz_gel model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_gauss_lorentz_gel component, generated from gauss_lorentz_gel.c in sasmodels.

Example: SasView_gauss_lorentz_gel(gauss_scale, cor_length_static, lorentz_scale, cor_length_dynamic, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_cor_length_static=0.0, pd_cor_length_dynamic=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
gauss_scale		([-inf, inf]) Gauss scale factor.	100.0
cor_length_static	Å	([0, inf]) Static correlation length.	100.0
lorentz_scale		([-inf, inf]) Lorentzian scale factor.	50.0
cor_length_dynamic	Å	([0, inf]) Dynamic correlation length. Optional parameters:	20.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01

Name	Unit	Description	Default
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_cor_length_static		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_cor_length_dynamic		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_gauss_lorentz_gel.comp`.

13.41. The `SasView_gaussian_peak` McStas Component

SasView gaussian_peak model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_gaussian_peak component, generated from gaussian_peak.c in sasmodels.

Example: SasView_gaussian_peak(peak_pos, sigma, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0,)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
peak_pos	1/Å	([-inf, inf]) Peak position.	0.05
sigma	1/Å	([0, inf]) Peak width (standard deviation). Optional parameters:	0.005
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0

Links

- Component source code found in file `SasView_gaussian_peak.comp`.

13.42. The `SasView_gel_fit` McStas Component

SasView gel_fit model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_gel_fit component, generated from gel_fit.c in sasmodels.
Example: SasView_gel_fit(guinier_scale, lorentz_scale, rg, fractal_dim, cor_length, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_rg=0.0, pd_cor_length=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
guinier_scale	cm^{-1}	([-inf, inf]) Guinier term scale.	1.7
lorentz_scale	cm^{-1}	([-inf, inf]) Lorentz term scale.	3.5
rg	Å	([2, inf]) Radius of gyration.	104.0
fractal_dim		([0, inf]) Fractal exponent.	2.0
cor_length	Å	([0, inf]) Correlation length. Optional parameters:	16.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0

Name	Unit	Description	Default
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_rg		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_cor_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_gel_fit.comp`.

13.43. The `SasView_guinier` McStas Component

SasView guinier model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_guinier component, generated from guinier.c in sasmodels.

Example: SasView_guinier(rg, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_rg=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
rg	Å	([-inf, inf]) Radius of Gyration. Optional parameters:	60.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_rg		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_guinier.comp`.

13.44. The `SasView_guinier_porod` McStas Component

SasView guinier_porod model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_guinier_porod component, generated from guinier_porod.c in sasmodels.
Example: SasView_guinier_porod(rg, s, porod_exp, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_rg=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
rg	Å	([0, inf]) Radius of gyration.	60.0
s		([0, inf]) Dimension variable.	1.0
porod_exp		([0, inf]) Porod exponent. Optional parameters:	3.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0

Name	Unit	Description	Default
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_rg		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_guinier_porod.comp`.

13.45. The `SasView_hardsphere` McStas Component

SasView hardsphere model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_hardsphere component, generated from hardsphere.c in sasmodels.

Example: SasView_hardsphere(radius_effective, volfraction, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_effective=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius_effective	Å	([0, inf]) effective radius of hard sphere.	50.0
volfraction		([0, 0.74]) volume fraction of hard spheres. Optional parameters:	0.2
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius_effective		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_hardsphere.comp`.

13.46. The `SasView_hayter_msa` McStas Component

SasView hayter_msa model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_hayter_msa component, generated from hayter_msa.c in sasmodels.

Example: SasView_hayter_msa(radius_effective, volfraction, charge, temperature, concentration_salt, dielectconst, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_effective=0.0, pd_charge=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius_effective	Å	([0, inf]) effective radius of charged sphere.	20.75
volfraction	None	([0, 0.74]) volume fraction of spheres.	0.0192
charge	e	([1e-06, 200]) charge on sphere (in electrons).	19.0
temperature	K	([0, 450]) temperature, in Kelvin, for Debye length calculation.	318.16
concentration_salt	M	([0, inf]) conc of salt, moles/litre, 1:1 electolyte, for Debye length.	0.0
dielectconst	None	([-inf, inf]) dielectric constant (relative permittivity) of solvent, kappa, default water, for Debye length. Optional parameters:	71.08
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01

Name	Unit	Description	Default
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius_effective		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_charge		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_hayter_msa.comp`.

13.47. The `SasView_hollow_cylinder` McStas Component

SasView hollow_cylinder model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_hollow_cylinder component, generated from hollow_cylinder.c in sasmodels.

Example: SasView_hollow_cylinder(radius, thickness, length, sld, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_thickness=0.0, pd_length=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	Å	([0, inf]) Cylinder core radius.	20.0
thickness	Å	([0, inf]) Cylinder wall thickness.	10.0
length	Å	([0, inf]) Cylinder total length.	400.0
sld	1e-6/Å^2	([-inf, inf]) Cylinder sld.	6.3
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent sld. Optional parameters:	1
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0

Name	Unit	Description	Default
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thickness		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_hollow_cylinder.comp`.

13.48. The `SasView_hollow_cylinder_aniso` McStas Component

SasView hollow_cylinder model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_hollow_cylinder component, generated from hollow_cylinder.c in sasmodels.
Example: SasView_hollow_cylinder_aniso(radius, thickness, length, sld, sld_solvent, theta, phi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_thickness=0.0, pd_length=0.0, pd_theta=0.0, pd_phi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	Å	([0, inf]) Cylinder core radius.	20.0

Name	Unit	Description	Default
thickness	Å	([0, inf]) Cylinder wall thickness.	10.0
length	Å	([0, inf]) Cylinder total length.	400.0
sld	1e-6/Å^2	([-inf, inf]) Cylinder sld.	6.3
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent sld. Optional parameters:	1
theta			90
phi			0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thickness		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0

Name	Unit	Description	Default
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_hollow_cylinder_aniso.comp`.

13.49. The `SasView_hollow_rectangular_prism` McStas Component

SasView hollow_rectangular_prism model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_hollow_rectangular_prism component, generated from hollow_rectangular_prism.c in sasmodels.

Example: SasView_hollow_rectangular_prism(sld, sld_solvent, length_a, b2a_ratio, c2a_ratio, thickness, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_length_a=0.0, pd_thickness=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Parallelepiped scattering length density.	6.3

Name	Unit	Description	Default
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1
length_a	Å	([0, inf]) Shortest, external, size of the parallelepiped.	35
b2a_ratio	Å	([0, inf]) Ratio sides b/a.	1
c2a_ratio	Å	([0, inf]) Ratio sides c/a.	1
thickness	Å	([0, inf]) Thickness of parallelepiped. Optional parameters:	1
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_length_a		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thickness		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_hollow_rectangular_prism.comp`.

13.50. The `SasView_hollow_rectangular_prism_aniso` McStas Component

SasView hollow_rectangular_prism model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_hollow_rectangular_prism component, generated from hollow_rectangular_prism.c in sasmodels.

Example: SasView_hollow_rectangular_prism_aniso(sld, sld_solvent, length_a, b2a_ratio, c2a_ratio, thickness, theta, phi, Psi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_length_a=0.0, pd_thickness=0.0, pd_theta=0.0, pd_phi=0.0, pd_Psi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Parallelepiped scattering length density.	6.3
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1
length_a	Å	([0, inf]) Shortest, external, size of the parallelepiped.	35
b2a_ratio	Å	([0, inf]) Ratio sides b/a.	1
c2a_ratio	Å	([0, inf]) Ratio sides c/a.	1
thickness	Å	([0, inf]) Thickness of parallelepiped. Optional parameters:	1
theta			0
phi			0
Psi			0

Name	Unit	Description	Default
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_length_a		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thickness		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_Psi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_hollow_rectangular_prism_aniso.comp`.

13.51. The `SasView_hollow_rectangular_prism_thin_walls` McStas Component

SasView hollow_rectangular_prism_thin_walls model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_hollow_rectangular_prism_thin_walls component, generated from hollow_rectangular_prism_thin_w
in sasmodels.

Example: SasView_hollow_rectangular_prism_thin_walls(sld, sld_solvent, length_a, b2a_ratio, c2a_ratio, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_length_a=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Parallelepiped scattering length density.	6.3
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1
length_a	Å	([0, inf]) Shorter side of the parallelepiped.	35
b2a_ratio	Å	([0, inf]) Ratio sides b/a.	1
c2a_ratio	Å	([0, inf]) Ratio sides c/a. Optional parameters:	1

Name	Unit	Description	Default
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_length_a		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_hollow_rectangular_prism_thin_walls.comp`.

13.52. The `SasView_lamellar_hg` McStas Component

SasView lamellar_hg model component as sample description.

Identification

- **Author:** Jose Robledo

- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_lamellar_hg component, generated from lamellar_hg.c in sasmodels.
Example: SasView_lamellar_hg(length_tail, length_head, sld, sld_head, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_length_tail=0.0, pd_length_head=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
length_tail	Å	([0, inf]) Tail thickness (total = H+T+T+H).	15
length_head	Å	([0, inf]) Head thickness.	10
sld	1e-6/Å^2	([-inf, inf]) Tail scattering length density.	0.4
sld_head	1e-6/Å^2	([-inf, inf]) Head scattering length density.	3.0
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	6
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1

Name	Unit	Description	Default
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_length_tail		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length_head		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_lamellar_hg.comp`.

13.53. The `SasView_lamellar_hg_stack_caille` McStas Component

SasView lamellar_hg_stack_caille model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_lamellar_hg_stack_caille component, generated from lamellar_hg_stack_caille.c in sasmodels.

Example: SasView_lamellar_hg_stack_caille(length_tail, length_head, Nlayers, d_spacing, Caille_parameter, sld, sld_head, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_length_tail=0.0 pd_length_head=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
length_tail	Å	([0, inf]) Tail thickness.	10
length_head	Å	([0, inf]) head thickness.	2
Nlayers		([1, inf]) Number of layers.	30
d_spacing	Å	([0.0, inf]) lamellar d-spacing of Caille S(Q).	40.0
Caille_parameter		([0.0, 0.8]) Caille parameter.	0.001
sld	1e-6/Å^2	([-inf, inf]) Tail scattering length density.	0.4
sld_head	1e-6/Å^2	([-inf, inf]) Head scattering length density.	2.0
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	6
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0

Name	Unit	Description	Default
pd_length_tail		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length_head		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_lamellar_hg_stack_caille.comp`.

13.54. The `SasView_lamellar_stack_caille` McStas Component

SasView lamellar_stack_caille model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_lamellar_stack_caille component, generated from lamellar_stack_caille.c in sasmodels.

Example: SasView_lamellar_stack_caille(thickness, Nlayers, d_spacing, Caille_parameter, sld, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_thickness=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
thickness	Å	([0, inf]) sheet thickness.	30.0
Nlayers		([1, inf]) Number of layers.	20
d_spacing	Å	([0.0, inf]) lamellar d-spacing of Caille S(Q).	400.0

Name	Unit	Description	Default
Caille_parameter	1/Å^2	([0.0, 0.8]) Caille parameter.	0.1
sld	1e-6/Å^2	([-inf, inf]) layer scattering length density.	6.3
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	1.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_thickness		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_lamellar_stack_caille.comp`.

13.55. The `SasView_lamellar_stack_paracrystal` McStas Component

SasView lamellar_stack_paracrystal model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_lamellar_stack_paracrystal component, generated from lamellar_stack_paracrystal.c in sasmodels.

Example: SasView_lamellar_stack_paracrystal(thickness, Nlayers, d_spacing, sigma_d, sld, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_thickness=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
thickness	Å	([0, inf]) sheet thickness.	33.0
Nlayers		([1, inf]) Number of layers.	20
d_spacing	Å	([0.0, inf]) lamellar spacing of paracrystal stack.	250.0
sigma_d	Å	([0.0, inf]) Sigma (polydispersity) of the lamellar spacing.	0.0
sld	1e-6/Å^2	([-inf, inf]) layer scattering length density.	1.0
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	6.34
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0

Name	Unit	Description	Default
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_thickness		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_lamellar_stack_paracrystal.comp`.

13.56. The `SasView_line` McStas Component

SasView line model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_line component, generated from line.c in sasmodels.

Example: SasView_line(intercept, slope, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0,)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
intercept	1/cm	([-inf, inf]) intercept in linear model.	1.0
slope	Å/cm	([-inf, inf]) slope in linear model. Optional parameters:	1.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0

Links

- Component source code found in file `SasView_line.comp`.

13.57. The `SasView_linear_pearls` McStas Component

SasView linear_pearls model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_linear_pearls component, generated from linear_pearls.c in sasmodels.
Example: SasView_linear_pearls(radius, edge_sep, num_pearls, sld, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	Å	([0, inf]) Radius of the pearls.	80.0
edge_sep	Å	([0, inf]) Length of the string segment - surface to surface.	350.0
num_pearls		([1, inf]) Number of the pearls.	3.0
sld	1e-6/Å^2	([-inf, inf]) SLD of the pearl spheres.	1.0
sld_solvent	1e-6/Å^2	([-inf, inf]) SLD of the solvent. Optional parameters:	6.3
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005

Name	Unit	Description	Default
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_linear_pearls.comp`.

13.58. The `SasView_lorentz` McStas Component

SasView lorentz model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_lorentz component, generated from lorentz.c in sasmodels.

Example: SasView_lorentz(cor_length, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_cor_length=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
cor_length	Å	([0, inf]) Screening length. Optional parameters:	50.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_cor_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_lorentz.comp`.

13.59. The `SasView_mass_fractal` McStas Component

SasView mass_fractal model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_mass_fractal component, generated from mass_fractal.c in sasmodels.
Example: SasView_mass_fractal(radius, fractal_dim_mass, cutoff_length, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_cutoff_length=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	Å	([0.0, inf]) Particle radius.	10.0
fractal_dim_mass		([1.0, 6.0]) Mass fractal dimension.	1.9
cutoff_length	Å	([0.0, inf]) Cut-off length. Optional parameters:	100.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0

Name	Unit	Description	Default
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_cutoff_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_mass_fractal.comp`.

13.60. The `SasView_mass_surface_fractal` McStas Component

SasView mass_surface_fractal model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_mass_surface_fractal component, generated from mass_surface_fractal.c in sasmodels.

Example: SasView_mass_surface_fractal(fractal_dim_mass, fractal_dim_surf, rg_cluster, rg_primary, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_rg_cluster=0.0, pd_rg_primary=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
fractal_dim_mass		([0.0, 6.0]) Mass fractal dimension.	1.8
fractal_dim_surf		([0.0, 6.0]) Surface fractal dimension.	2.3
rg_cluster	Å	([0.0, inf]) Cluster radius of gyration.	4000.0
rg_primary	Å	([0.0, inf]) Primary particle radius of gyration. Optional parameters:	86.7
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_rg_cluster		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_rg_primary		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_mass_surface_fractal.comp`.

13.61. The `SasView_mono_gauss_coil` McStas Component

SasView mono_gauss_coil model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_mono_gauss_coil component, generated from mono_gauss_coil.c in sasmodels.

Example: SasView_mono_gauss_coil(i_zero, rg, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_rg=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
i_zero	1/cm	([0.0, inf]) Intensity at q=0.	70.0
rg	Å	([0.0, inf]) Radius of gyration. Optional parameters:	75.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005

Name	Unit	Description	Default
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_rg		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_mono_gauss_coil.comp`.

13.62. The `SasView_multilayer_vesicle` McStas Component

SasView multilayer_vesicle model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_multilayer_vesicle component, generated from multilayer_vesicle.c in sasmodels.

Example: SasView_multilayer_vesicle(volfraction, radius, thick_shell, thick_solvent, sld_solvent, sld, n_shells, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_thick_shell=
pd_thick_solvent=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
volfraction		([0.0, 1]) volume fraction of vesicles.	0.05
radius	Å	([0.0, inf]) radius of solvent filled core.	60.0
thick_shell	Å	([0.0, inf]) thickness of one shell.	10.0
thick_solvent	Å	([0.0, inf]) solvent thickness between shells.	10.0
sld_solvent	1e-6/Å^2	([-inf, inf]) solvent scattering length density.	6.4
sld	1e-6/Å^2	([-inf, inf]) Shell scattering length density.	0.4
n_shells		([1.0, inf]) Number of shell plus solvent layer pairs (must be integer). Optional parameters:	2.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0

Name	Unit	Description	Default
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_shell		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_solvent		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_multilayer_vesicle.comp`.

13.63. The `SasView_onion` McStas Component

SasView onion model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_onion component, generated from onion.c in sasmodels.

Example: SasView_onion(sld_core, radius_core, sld_solvent, n_shells, sld_in[n_shells], sld_out[n_shells], thickness[n_shells], A[n_shells], model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_core=0.0, pd_thickness[n_shells]=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default

Links

- Component source code found in file `SasView_onion.comp`.

13.64. The `SasView_parallelepiped` McStas Component

SasView parallelepiped model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_parallelepiped component, generated from parallelepiped.c in sasmodels.
Example: SasView_parallelepiped(sld, sld_solvent, length_a, length_b, length_c, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_length_a=0.0, pd_length_b=0.0, pd_length_c=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Parallelepiped scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1
length_a	Å	([0, inf]) Shorter side of the parallelepiped.	35
length_b	Å	([0, inf]) Second side of the parallelepiped.	75
length_c	Å	([0, inf]) Larger side of the parallelepiped. Optional parameters:	400
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0

Name	Unit	Description	Default
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_length_a		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length_b		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length_c		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_parallelepiped.comp`.

13.65. The `SasView_parallelepiped_aniso` McStas Component

SasView parallelepiped model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_parallelepiped component, generated from parallelepiped.c in sasmodels.

Example: SasView_parallelepiped_aniso(sld, sld_solvent, length_a, length_b, length_c, theta, phi, Psi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_length_a=0.0, pd_length_b=0.0, pd_length_c=0.0, pd_theta=0.0, pd_phi=0.0, pd_Psi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Parallelepiped scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1
length_a	Å	([0, inf]) Shorter side of the parallelepiped.	35
length_b	Å	([0, inf]) Second side of the parallelepiped.	75
length_c	Å	([0, inf]) Larger side of the parallelepiped. Optional parameters:	400
theta			60
phi			60
Psi			60
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01

Name	Unit	Description	Default
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_length_a		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length_b		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_length_c		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_Psi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_parallelepiped_aniso.comp`.

13.66. The `SasView_peak_lorentz` McStas Component

SasView peak_lorentz model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_peak_lorentz component, generated from peak_lorentz.c in sasmodels.
Example: SasView_peak_lorentz(peak_pos, peak_hwhm, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0,)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
peak_pos	1/Å	([-inf, inf]) Peak postion in q.	0.05
peak_hwhm	1/Å	([-inf, inf]) HWHM of peak. Optional parameters:	0.005
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1

Name	Unit	Description	Default
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0

Links

- Component source code found in file `SasView_peak_lorentz.comp`.

13.67. The `SasView_pearl_necklace` McStas Component

SasView pearl_necklace model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_pearl_necklace component, generated from pearl_necklace.c in sasmodels.
Example: SasView_pearl_necklace(radius, edge_sep, thick_string, num_pearls, sld, sld_string, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_thick_string

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	Å	([0, inf]) Mean radius of the chained spheres.	80.0
edge_sep	Å	([0, inf]) Mean separation of chained particles.	350.0

Name	Unit	Description	Default
thick_string	Å	([0, inf]) Thickness of the chain linkage.	2.5
num_pearls	none	([1, inf]) Number of pearls in the necklace (must be integer).	3
sld	1e-6/Å^2	([-inf, inf]) Scattering length density of the chained spheres.	1.0
sld_string	1e-6/Å^2	([-inf, inf]) Scattering length density of the chain linkage.	1.0
sld_solvent	1e-6/Å^2	([-inf, inf]) Scattering length density of the solvent. Optional parameters:	6.3
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_string		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_pearl_necklace.comp`.

13.68. The `SasView_poly_gauss_coil` McStas Component

SasView poly_gauss_coil model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_poly_gauss_coil component, generated from poly_gauss_coil.c in sasmodels.
Example: SasView_poly_gauss_coil(i_zero, rg, polydispersity, model_scale=1.0, model_abs=0.0 xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_rg=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
i_zero	1/cm	([0.0, inf]) Intensity at q=0.	70.0
rg	Å	([0.0, inf]) Radius of gyration.	75.0
polydispersity	None	([1.0, inf]) Polymer Mw/Mn. Optional parameters:	2.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005

Name	Unit	Description	Default
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_rg		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_poly_gauss_coil.comp`.

13.69. The `SasView_polymer_excl_volume` McStas Component

SasView polymer_excl_volume model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_polymer_excl_volume component, generated from polymer_excl_volume.c in sasmodels.

Example: SasView_polymer_excl_volume(rg, porod_exp, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_rg=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
rg	Å	([0, inf]) Radius of Gyration.	60.0
porod_exp		([0, inf]) Porod exponent. Optional parameters:	3.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_rg		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_polymer_excl_volume.comp`.

13.70. The `SasView_polymer_micelle` McStas Component

SasView polymer_micelle model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_polymer_micelle component, generated from polymer_micelle.c in sasmodels.
Example: SasView_polymer_micelle(ndensity, v_core, v_corona, sld_solvent, sld_core, sld_corona, radius_core, rg, d_penetration, n_aggreg, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_core=0.0, pd_rg=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
ndensity	1e15/cm^3	([0.0, inf]) Number density of micelles.	8.94
v_core	Å^3	([0.0, inf]) Core volume .	62624.0
v_corona	Å^3	([0.0, inf]) Corona volume.	61940.0
sld_solvent	1e-6/Å^2	([0.0, inf]) Solvent scattering length density.	6.4
sld_core	1e-6/Å^2	([0.0, inf]) Core scattering length density.	0.34
sld_corona	1e-6/Å^2	([0.0, inf]) Corona scattering length density.	0.8
radius_core	Å	([0.0, inf]) Radius of core (must be >> rg).	45.0
rg	Å	([0.0, inf]) Radius of gyration of chains in corona.	20.0
d_penetration		([-inf, inf]) Factor to mimic non-penetration of Gaussian chains.	1.0
n_aggreg		([-inf, inf]) Aggregation number of the micelle. Optional parameters:	6.0

Name	Unit	Description	Default
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius_core		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_rg		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_polymer_micelle.comp`.

13.71. The `SasView_porod` McStas Component

SasView porod model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_porod component, generated from porod.c in sasmodels.
Example: SasView_porod(, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0,)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0

Name	Unit	Description	Default
focus_r	m	case of circular focusing, focusing radius.	0

Links

- Component source code found in file `SasView_porod.comp`.

13.72. The `SasView_power_law` McStas Component

SasView power_law model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_power_law component, generated from power_law.c in sasmodels.
Example: SasView_power_law(power, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0,)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
power		([-inf, inf]) Power law exponent. Optional parameters:	4.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01

Name	Unit	Description	Default
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0

Links

- Component source code found in file `SasView_power_law.comp`.

13.73. The `SasView_pringle` McStas Component

SasView pringle model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_pringle component, generated from pringle.c in sasmodels.

Example: SasView_pringle(radius, thickness, alpha, beta, sld, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_thickness=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	Å	([0, inf]) Pringle radius.	60.0
thickness	Å	([0, inf]) Thickness of pringle.	10.0
alpha		([-inf, inf]) Curvature parameter alpha.	0.001
beta		([-inf, inf]) Curvature paramter beta.	0.02
sld	1e-6/Å^2	([-inf, inf]) Pringle sld.	1.0
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent sld. Optional parameters:	6.3
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thickness		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_pringle.comp`.

13.74. The `SasView_raspberry` McStas Component

SasView raspberry model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_raspberry component, generated from raspberry.c in sasmodels.
Example: SasView_raspberry(sld_lg, sld_sm, sld_solvent, volfraction_lg, volfraction_sm, surface_fraction, radius_lg, radius_sm, penetration, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_lg=0.0, pd_radius_sm=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld_lg	1e-6/Å^2	([-inf, inf]) large particle scattering length density.	-0.4
sld_sm	1e-6/Å^2	([-inf, inf]) small particle scattering length density.	3.5
sld_solvent	1e-6/Å^2	([-inf, inf]) solvent scattering length density.	6.36
volfraction_lg		([-inf, inf]) volume fraction of large spheres.	0.05
volfraction_sm		([-inf, inf]) volume fraction of small spheres.	0.005
surface_fraction		([-inf, inf]) fraction of small spheres at surface.	0.4
radius_lg	Å	([0, inf]) radius of large spheres.	5000
radius_sm	Å	([0, inf]) radius of small spheres.	100

Name	Unit	Description	Default
penetration	Å	([-1, 1]) fractional penetration depth of small spheres into large sphere. Optional parameters:	0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius_lg		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_radius_sm		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_raspberry.comp`.

13.75. The `SasView_rectangular_prism` McStas Component

SasView rectangular_prism model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld			6.3
sld_solvent			1
length_a			35
b2a_ratio			1
c2a_ratio			1
model_scale			1.0
model_abs			0.0
xwidth			0.01
yheight			0.01
zdepth			0.005
R			0
target_x			0
target_y			0
target_z			1
target_index			1
focus_xw			0.5
focus_yh			0.5
focus_aw			0
focus_ah			0
focus_r			0
pd_length_a			0.0

Links

- Component source code found in file `SasView_rectangular_prism.comp`.

13.76. The SasView_rectangular_prism_aniso McStas Component

SasView rectangular_prism model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_rectangular_prism component, generated from rectangular_prism.c in sasmodels.

Example: SasView_rectangular_prism_aniso(sld, sld_solvent, length_a, b2a_ratio, c2a_ratio, theta, phi, Psi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_length_a=0.0, pd_theta=0.0, pd_phi=0.0, pd_Psi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Parallelepiped scattering length density.	6.3
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1
length_a	Å	([0, inf]) Shorter side of the parallelepiped.	35
b2a_ratio		([0, inf]) Ratio sides b/a.	1
c2a_ratio		([0, inf]) Ratio sides c/a. Optional parameters:	1
theta			0
phi			0
Psi			0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0

Name	Unit	Description	Default
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_length_a		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_Psi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_rectangular_prism_aniso.comp`.

13.77. The `SasView_rpa` McStas Component

SasView rpa model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_rpa component, generated from rpa.c in sasmodels.
Example:

```
SasView_rpa(case_num, N[4], Phi[4], v[4], L[4], b[4], K12, K13, K14, K23, K24, K34,
```

model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0,)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default

Links

- Component source code found in file `SasView_rpa.comp`.

13.78. The `SasView_sc_paracrystal` McStas Component

SasView sc_paracrystal model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_sc_paracrystal component, generated from sc_paracrystal.c in sasmodels.
Example: SasView_sc_paracrystal(dnn, d_factor, radius, sld, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
dnn	Å	([0.0, inf]) Nearest neighbor distance.	220.0
d_factor		([-inf, inf]) Paracrystal distortion factor.	0.06
radius	Å	([0.0, inf]) Radius of sphere.	40.0
sld	1e-6/Å^2	([0.0, inf]) Sphere scattering length density.	3.0
sld_solvent	1e-6/Å^2	([0.0, inf]) Solvent scattering length density. Optional parameters:	6.3
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_sc_paracrystal.comp`.

13.79. The `SasView_sc_paracrystal_aniso` McStas Component

SasView sc_paracrystal model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_sc_paracrystal component, generated from sc_paracrystal.c in sasmodels.
Example: SasView_sc_paracrystal_aniso(dnn, d_factor, radius, sld, sld_solvent, theta, phi, Psi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_theta=0.0, pd_phi=0.0, pd_Psi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
dnn	Å	([0.0, inf]) Nearest neighbor distance.	220.0
d_factor		([-inf, inf]) Paracrystal distortion factor.	0.06
radius	Å	([0.0, inf]) Radius of sphere.	40.0
sld	1e-6/Å^2	([0.0, inf]) Sphere scattering length density.	3.0
sld_solvent	1e-6/Å^2	([0.0, inf]) Solvent scattering length density. Optional parameters:	6.3
theta			0
phi			0
Psi			0

Name	Unit	Description	Default
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_Psi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_sc_paracrystal_aniso.comp`.

13.80. The `SasView_sphere` McStas Component

SasView sphere model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_sphere component, generated from sphere.c in sasmodels.
Example: SasView_sphere(sld, sld_solvent, radius, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Layer scattering length density.	1
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	6
radius	Å	([0, inf]) Sphere radius. Optional parameters:	50
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0

Name	Unit	Description	Default
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_sphere.comp`.

13.81. The `SasView_spinodal` McStas Component

SasView spinodal model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_spinodal component, generated from spinodal.c in sasmodels.

Example: SasView_spinodal(gamma, q_0, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0,)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
gamma		([-inf, inf]) Exponent.	3.0
q_0	1/Å	([-inf, inf]) Correlation peak position. Optional parameters:	0.1
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0

Links

- Component source code found in file `SasView_spinodal.comp`.

13.82. The `SasView_squarewell` McStas Component

SasView squarewell model component as sample description.

Identification

- **Author:** Jose Robledo

- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_squarewell component, generated from squarewell.c in sasmodels.

Example: SasView_squarewell(radius_effective, volfraction, welldepth, wellwidth, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_effective=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius_effective	Å	([0, inf]) effective radius of hard sphere.	50.0
volfraction		([0, 0.08]) volume fraction of spheres.	0.04
welldepth	kT	([0.0, 1.5]) depth of well, epsilon.	1.5
wellwidth	diameters	([1.0, inf]) width of well in diameters (=2R) units, must be > 1. Optional parameters:	1.2
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5

Name	Unit	Description	Default
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius_effective		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_squarewell.comp`.

13.83. The `SasView_stacked_disks` McStas Component

SasView stacked_disks model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_stacked_disks component, generated from stacked_disks.c in sasmodels.

Example: SasView_stacked_disks(thick_core, thick_layer, radius, n_stacking, sigma_d, sld_core, sld_layer, sld_solvent, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_thick_core=0.0, pd_thick_layer=0.0, pd_radius=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
thick_core	Å	([0, inf]) Thickness of the core disk.	10.0
thick_layer	Å	([0, inf]) Thickness of layer each side of core.	10.0
radius	Å	([0, inf]) Radius of the stacked disk.	15.0

Name	Unit	Description	Default
n_stacking		([1, inf]) Number of stacked layer/core/layer disks.	1.0
sigma_d	Å	([0, inf]) Sigma of nearest neighbor spacing.	0
sld_core	1e-6/Å^2	([-inf, inf]) Core scattering length density.	4
sld_layer	1e-6/Å^2	([-inf, inf]) Layer scattering length density.	0.0
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	5.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_thick_core		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_layer		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0

Name	Unit	Description	Default
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_stacked_disks.comp`.

13.84. The `SasView_stacked_disks_aniso` McStas Component

SasView stacked_disks model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_stacked_disks component, generated from stacked_disks.c in sasmodels.
Example: SasView_stacked_disks_aniso(thick_core, thick_layer, radius, n_stacking, sigma_d, sld_core, sld_layer, sld_solvent, theta, phi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_thick_core=0.0, pd_thick_layer=0.0, pd_radius=0.0, pd_theta=0.0, pd_phi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
thick_core	Å	([0, inf]) Thickness of the core disk.	10.0
thick_layer	Å	([0, inf]) Thickness of layer each side of core.	10.0
radius	Å	([0, inf]) Radius of the stacked disk.	15.0
n_stacking		([1, inf]) Number of stacked layer/core/layer disks.	1.0

Name	Unit	Description	Default
sigma_d	Å	([0, inf]) Sigma of nearest neighbor spacing.	0
sld_core	1e-6/Å^2	([-inf, inf]) Core scattering length density.	4
sld_layer	1e-6/Å^2	([-inf, inf]) Layer scattering length density.	0.0
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density. Optional parameters:	5.0
theta			0
phi			0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_thick_core		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thick_layer		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0

Name	Unit	Description	Default
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_stacked_disks_aniso.comp`.

13.85. The `SasView_star_polymer` McStas Component

SasView star_polymer model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_star_polymer component, generated from star_polymer.c in sasmodels.
Example: SasView_star_polymer(rg_squared, arms, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_rg_squared=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
rg_squared	Å^2	([0.0, inf]) Ensemble radius of gyration SQUARED of the full polymer.	100.0
arms		([1.0, 6.0]) Number of arms in the model. Optional parameters:	3

Name	Unit	Description	Default
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_rg_squared		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_star_polymer.comp`.

13.86. The `SasView_stickyhardsphere` McStas Component

SasView stickyhardsphere model component as sample description.

Identification

- **Author:** Jose Robledo

- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_stickyhardsphere component, generated from stickyhardsphere.c in sasmodels.
Example: SasView_stickyhardsphere(radius_effective, volfraction, perturb, stickiness, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_effective=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius_effective	Å	([0, inf]) effective radius of hard sphere.	50.0
volfraction		([0, 0.74]) volume fraction of hard spheres.	0.2
perturb		([0.01, 0.1]) perturbation parameter, tau.	0.05
stickiness		([-inf, inf]) stickiness, epsilon. Optional parameters:	0.2
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5

Name	Unit	Description	Default
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius_effective		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_stickyhardsphere.comp`.

13.87. The `SasView_superball` McStas Component

SasView superball model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_superball component, generated from superball.c in sasmodels.

Example: SasView_superball(sld, sld_solvent, length_a, exponent_p, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_length_a=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Superball scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1

Name	Unit	Description	Default
length_a	Å	([0, inf]) Cube edge length of the superball.	50
exponent_p		([0, inf]) Exponent describing the roundness of the superball. Optional parameters:	2.5
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_length_a		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_superball.comp`.

13.88. The `SasView_superball_aniso` McStas Component

SasView superball model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_superball component, generated from superball.c in sasmodels.
Example: SasView_superball_aniso(sld, sld_solvent, length_a, exponent_p, theta, phi, Psi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_length_a=0.0, pd_theta=0.0, pd_phi=0.0, pd_Psi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Superball scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1
length_a	Å	([0, inf]) Cube edge length of the superball.	50
exponent_p		([0, inf]) Exponent describing the roundness of the superball. Optional parameters:	2.5
theta			0
phi			0
Psi			0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01

Name	Unit	Description	Default
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_length_a		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_Psi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_superball_aniso.comp`.

13.89. The `SasView_surface_fractal` McStas Component

SasView surface_fractal model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_surface_fractal component, generated from surface_fractal.c in sasmodels.

Example: SasView_surface_fractal(radius, fractal_dim_surf, cutoff_length, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_cutoff_length=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	Å	([0, inf]) Particle radius.	10.0
fractal_dim_surf		([1, 3]) Surface fractal dimension.	2.0
cutoff_length	Å	([0.0, inf]) Cut-off Length. Optional parameters:	500.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0

Name	Unit	Description	Default
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_cutoff_length		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_surface_fractal.comp`.

13.90. The `SasView_teubner_strey` McStas Component

SasView teubner_strey model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_teubner_strey component, generated from teubner_strey.c in sasmodels.
Example: SasView_teubner_strey(volfraction_a, sld_a, sld_b, d, xi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0,)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
volfraction_a		([0, 1.0]) Volume fraction of phase a.	0.5
sld_a	1e-6/Å^2	([-inf, inf]) SLD of phase a.	0.3
sld_b	1e-6/Å^2	([-inf, inf]) SLD of phase b.	6.3
d	Å	([0, inf]) Domain size (periodicity).	100.0
xi	Å	([0, inf]) Correlation length. Optional parameters:	30.0

Name	Unit	Description	Default
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0

Links

- Component source code found in file `SasView_teubner_strey.comp`.

13.91. The `SasView_triaxial_ellipsoid` McStas Component

SasView triaxial_ellipsoid model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_triaxial_ellipsoid component, generated from triaxial_ellipsoid.c in sasmodels.

Example: SasView_triaxial_ellipsoid(sld, sld_solvent, radius_equat_minor, radius_equat_major radius_polar, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_equat_minor=0.0, pd_radius_equa pd_radius_polar=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Ellipsoid scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1
radius_equat_minor	Å	([0, inf]) Minor equatorial radius, Ra.	20
radius_equat_major	Å	([0, inf]) Major equatorial radius, Rb.	400
radius_polar	Å	([0, inf]) Polar radius, Rc. Optional parameters:	10
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5

Name	Unit	Description	Default
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_radius_equat_minor		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_radius_equat_major		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_radius_polar		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_triaxial_ellipsoid.comp`.

13.92. The `SasView_triaxial_ellipsoid_aniso` McStas Component

SasView triaxial_ellipsoid model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_triaxial_ellipsoid component, generated from triaxial_ellipsoid.c in sasmodels.

Example: SasView_triaxial_ellipsoid_aniso(sld, sld_solvent, radius_equat_minor, radius_equat_major, radius_polar, theta, phi, Psi, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius_equat_minor=0.0, pd_radius_equat_major=0.0, pd_radius_polar=0.0, pd_theta=0.0, pd_phi=0.0, pd_Psi=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) Ellipsoid scattering length density.	4
sld_solvent	1e-6/Å^2	([-inf, inf]) Solvent scattering length density.	1
radius_equat_minor	Å	([0, inf]) Minor equatorial radius, Ra.	20
radius_equat_major	Å	([0, inf]) Major equatorial radius, Rb.	400
radius_polar	Å	([0, inf]) Polar radius, Rc. Optional parameters:	10
theta			60
phi			60
Psi			60
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0

Name	Unit	Description	Default
pd_radius_equat_minor		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_radius_equat_major		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_radius_polar		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_theta		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_phi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_Psi		(0,360) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_triaxial_ellipsoid_aniso.comp`.

13.93. The `SasView_two_lorentzian` McStas Component

SasView two_lorentzian model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_two_lorentzian component, generated from two_lorentzian.c in sasmodels.

Example: SasView_two_lorentzian(lorentz_scale_1, lorentz_length_1, lorentz_exp_1, lorentz_scale_2, lorentz_length_2, lorentz_exp_2, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_lorentz_length_1=0.0, pd_lorentz_length_2=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
lorentz_scale_1		([-inf, inf]) First power law scale factor.	10.0
lorentz_length_1	Å	([-inf, inf]) First Lorentzian screening length.	100.0
lorentz_exp_1		([-inf, inf]) First exponent of power law.	3.0
lorentz_scale_2		([-inf, inf]) Second scale factor for broad Lorentzian peak.	1.0
lorentz_length_2	Å	([-inf, inf]) Second Lorentzian screening length.	10.0
lorentz_exp_2		([-inf, inf]) Second exponent of power law. Optional parameters:	2.0
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0
pd_lorentz_length_1		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0

Name	Unit	Description	Default
pd_lorentz_length_2		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_two_lorentzian.comp`.

13.94. The `SasView_two_power_law` McStas Component

SasView two_power_law model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_two_power_law component, generated from two_power_law.c in sasmodels.
Example: SasView_two_power_law(coefficent_1, crossover, power_1, power_2, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1, target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0,)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
coefficent_1		([-inf, inf]) coefficent A in low Q region.	1.0
crossover	1/Å	([0, inf]) crossover location.	0.04
power_1		([0, inf]) power law exponent at low Q.	1.0
power_2		([0, inf]) power law exponent at high Q.	4.0
		Optional parameters:	
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0

Name	Unit	Description	Default
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0

Links

- Component source code found in file `SasView_two_power_law.comp`.

13.95. The `SasView_vesicle` McStas Component

SasView vesicle model component as sample description.

Identification

- **Author:** Jose Robledo
- **Origin:** FZJ / DTU / ESS DMSC
- **Date:**

Description

SasView_vesicle component, generated from vesicle.c in sasmodels.

Example: SasView_vesicle(sld, sld_solvent, volfraction, radius, thickness, model_scale=1.0, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005, R=0, int target_index=1,

target_x=0, target_y=0, target_z=1, focus_xw=0.5, focus_yh=0.5, focus_aw=0, focus_ah=0, focus_r=0, pd_radius=0.0, pd_thickness=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sld	1e-6/Å^2	([-inf, inf]) vesicle shell scattering length density.	0.5
sld_solvent	1e-6/Å^2	([-inf, inf]) solvent scattering length density.	6.36
volfraction		([0, 1.0]) volume fraction of shell.	0.05
radius	Å	([0, inf]) vesicle core radius.	100
thickness	Å	([0, inf]) vesicle shell thickness. Optional parameters:	30
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_abs		Absorption cross section density at 2200 m/s.	0.0
xwidth	m	([-inf, inf]) Horiz. dimension of sample, as a width.	0.01
yheight	m	([-inf, inf]) vert . dimension of sample, as a height for cylinder/box	0.01
zdepth	m	([-inf, inf]) depth of sample	0.005
R	m	Outer radius of sample in (x,z) plane for cylinder/sphere.	0
target_x	m	relative focus target position.	0
target_y	m	relative focus target position.	0
target_z	m	relative focus target position.	1
target_index		Relative index of component to focus at, e.g. next is +1.	1
focus_xw	m	horiz. dimension of a rectangular area.	0.5
focus_yh	m	, vert. dimension of a rectangular area.	0.5
focus_aw	deg	, horiz. angular dimension of a rectangular area.	0
focus_ah	deg	, vert. angular dimension of a rectangular area.	0
focus_r	m	case of circular focusing, focusing radius.	0

Name	Unit	Description	Default
pd_radius		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable.	0.0
pd_thickness		(0,inf) defined as (dx/x), where x is de mean value and dx the standard devition of the variable	0.0

Links

- Component source code found in file `SasView_vesicle.comp`.

14. Contributed components

The `contrib` category holds components contributed directly by McStas users, rather than authored and maintained by the core McStas team. The McStas team provides the same technical support for these as for any other component, but the original authors – credited individually in each component's source header and reproduced below – carry primary responsibility for the underlying physics and its validation. Contributed components that mature and see wide use are periodically promoted into a core category (`sources`, `optics`, `samples`, `monitors`); several components documented in earlier chapters of this manual (e.g. several guide, chopper and polarisation components) began their life in `contrib`. Always check with `mcdoc` rather than an older document for a component's current category.

14.1. Contributed sources

14.2. The `ISIS_moderator` McStas Component

ISIS Moderators

Identification

- **Author:** S. Ansell and D. Champion
- **Origin:** ISIS
- **Date:** AUGUST 2005

Description

Produces a TS1 or TS2 ISIS moderator distribution. The Face argument determines which moderator is to be sampled. Each face uses a different Etable file (the location of which is determined by the MCTABLES environment variable). Neutrons are created having a range of energies determined by the Emin and Emax arguments. Trajectories are produced such that they pass through the moderator face (defined by modXsixe and yheight) and a focusing rectangle (defined by xh,focus_yh and dist). Please download the documentation for precise instructions for use.

```
Example:   ISIS_moderator(Face ="water", Emin = 49.0,Emax = 51.0, dist = 1.0, focus_xw = 0.01,
```

focus_yh = 0.01, xwidth = 0.074, yheight = 0.074, CAngle = 0.0,SAC= 1)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
Face	word	Name of the face - TS2=groove,hydrogen,narrow,broad - - TS1=Water,ch4,h2,merlin or instrument name eg maps, crisp etc. - TS2= W1-9 Emax-9	"hydrogen"
Emin	meV	Lower edge of energy distribution	49.0
Emax	meV	Upper edge of energy distribution	51.0
dist	m	Distance from source to the focusing rectangle	1.0
focus_xw	m	Width of focusing rectangle	0.01
focus_yh	m	Height of focusing rectangle	0.01
xwidth	m	Moderator vertical size	0.074
yheight	m	Moderator Horizontal size	0.074
CAngle	radians	Angle from the centre line	0.0
SAC	boolean	Apply solid angle correction or not (1/0)	1
Lmin	meV	Lower edge of wavelength distribution	0
Lmax	meV	Upper edge of wavelength distribution	0
target_index	1	relative index of component to focus at, e.g. next is +1 this is used to compute 'dist' automatically.	1
verbose	boolean	Echo status information at everu 1e5'th neutron	0

Links

- Component source code found in file `ISIS_moderator.comp`.
- Further information should be
- downloaded from the ISIS MC website.
- **/

ISIS pulsed moderators

Introduction

The following document describes the functions obtained for models of TS2 as described in Table 14.2:

target	3.4cm diameter tantalum clad tungsten
reflector	Be + D_2O (80:20) at 300K
Composite Moderator Coupled	H_2 + CH_4 Groove: 3x8.3̇ cm 26K solid-CH_4 Hydrogen: 12x11cm 22K liquid H_2
Poisoned Moderator Decoupled	solid-CH_4 26K Narrow: Gd poison at 2.4 cm - 8 vanes Broad: 3.3 cm – not fully decoupled
PreModerators	0.85 cm and 0.75 cm H_2O

Table 14.2.: Description of Models

TS1 model is from the tungsten target as currently installed and positioned. The model also includes the MERLIN moderator, this makes no significant difference to the other moderator faces.

Using the McStas Module

You MUST first set the environment variable 'MCTABLES' to be the full path of the directory containing the table files:

```
BASH: export MCTABLES=/usr/local/lib/mcstas/contrib/ISIS_tables/
TCSH: setenv MCTABLES /usr/local/lib/mcstas/contrib/ISIS_tables/
```

In Windows this can be done using the 'My Computer' properties and selecting the 'Advanced' tab and the Environment variables button. This can of course be overridden by placing the appropriate moderator (h.*face*) files in the working directory.

The module requires a set of variables listed in Table 14.3 and described below.

The *Face* variable determines the moderator surface that will be viewed. There are two types of *Face* variable: i) Views from the centre of each moderator face defined by the name of the moderator, for TS1: Water, H2, CH4, Merlin and TS2: Hydrogen, Groove, Narrow, Broad. ii) Views seen by each beamline, for TS1: Prisma, Maps, crisp etc. and for TS2: E1-E9 (East) and W1-W9 (West).

The McStas distribution includes some example moderator files for TS1 (water,h2,ch4) and TS2 (broad, narrow, hydrogen, groove), but others are available at `http://www.isis.rl.ac.uk/Computing/Software/MC/`, including instrument specific models.

Variables *E0* and *E1* define an energy window for sampled neutrons. This can be used to increase the statistical accuracy of chopper and mirrored instruments. However, *E0* and *E1* cannot be equal (although they can be close). By default these arguments select energy in meV, if negative values are given, selection will be in terms of Angstroms.

Variables *dist*, *xw* and *yh* are the three component which will determine the directional acceptance window. They define a rectangle with centre at (0,0,dist) from the moderator position and with width *xw* meters and height *yh* meters. The initial direction of all the

neutrons are chosen (randomly) to originate from a point on the moderator surface and along a vector, such that without obstruction (and gravitational effects), they would pass through the rectangle. This should be used as a directional guide. All the neutrons start from the surface of the moderator and will be diverted/absorbed if they encountered other components. The guide system can be turned off by setting *dist* to zero.

The *CAngle* variable is used to rotate the viewed direction of the moderator and reduces the effective solid angle of the moderator face. Currently it is only for the horizontal plane. This is redundant since there are beamline specific h.*face* files.

The two variables *modYsize* and *modXsize* allow the moderators to be effectively reduced/increased. If these variables are given negative or zero values then they default to the actual visible surface size of the moderators.

The last variable *SAC* will correct for the different solid angle seen by two focussing windows which are at different distances from the moderator surface. The normal measurement of flux is in neutrons/second/Å/cm^2/str, but in a detector it is measured in neutrons/second. Therefore if all other denominators in the flux are multiplied out then the flux at a point-sized focus window should follow an inverse square law. This solid angle correction is made if the *SAC* variable is set equal to 1, it will not be calculated if *SAC* is set to zero. It is advisable to select this variable at all times as it will give the most realistic results

Comparing TS1 and TS2

The Flux data provided in both sets of h.*face* files is for 60 μAmp sources. To compare TS1 and TS2, the TS1 data must be multiplied by three (current average strength of TS1 source 180 μAmps). When the 300 μAmp upgrade happens this factor should be revised accordingly.

Bugs

Sometimes if a particularly long wavelength ($>$ 20 Å) is requested there may be problems with sampling the data. In general the data used for long wavelengths should only be taken as a guide and not used for accurate simulations. At 9 Åthere is a kink in the distribution which is also to do with the MCNPX model changing. If this energy is sampled over then the results should be considered carefully.

14.3. The `ViewModISIS` McStas Component

ISIS Moderators

Identification

- **Author:** G. Skoro, based on ViewModerator4 from S. Ansell
- **Origin:** ISIS

Variable	Type	Options	Units	Description
Face (TS2)	char*	i) Hydrogen Groove Narrow Broad, ii) E1-E9 W1-W9	–	String which designates the name of the face
Face (TS1)	char*	i) H2 CH4 Merlin Water, ii) Maps Crisp Gem EVS HET HRPD Iris Mari Polaris Prisma Sandals Surf SXD Tosca	–	String which designates the name of the face
E0	float	0<E0<E1	meV (Å)	Only neutrons above this energy are sampled
E1	float	E0<E1<1e10	meV (Å)	Only neutrons below this energy are sampled
dist	float	$0 < dist < \infty$	m	Distance of focus window from face of moderator
xw	float	$0 < xw < \infty$	m	x width of the focus window
yh	float	$0 < yh < \infty$	m	y height of the focus window
CAngle	float	-360 $< CAngle <$ 360	^{o}	Horizontal angle from the normal to the moderator surface
modXsize	float	$0 < modXsize < \infty$	m	Horizontal size of the moderator (defaults to actual size)
modYsize	float	$0 < modYsize < \infty$	m	Vertical size of the moderator (defaults to actual size)
SAC	int	0,1	n/a	Solid Angle Correction

Table 14.3.: Brief Description of Variables

- **Date:** February 2016

Description

Produces a neutron distribution at the ISIS TS1 or TS2 beamline shutter front face (or corresponding moderator) position. The Face argument determines which TS1 or TS2 beamline is to be sampled by using corresponding Etable file. Neutrons are created having a range of energies determined by the E0 and E1 arguments. Trajectories are produced such that they pass through the shutter front face (RECOMENDED) or moderator face (defined by xw and yh) and a focusing rectangle (defined by focus_xw, focus_yh and dist).

```
Example 1:   ViewModISISver1(Face="TS1_S04_Merlin.mcstas", E0 = E_min, E1 = E_max,
```

dist = 1.7, focus_xw = 0.094, focus_yh = 0.094, modPosition=0,

```
xw=0.12,yh=0.12)
```

MERLIN simulation. modPosition=0 so initial surface is at (near to) the moderator face. In this example, focus_xw and focus_yh are chosen to be identical to the shutter opening dimension. dist = 1.7 is the 'real' distance to the shutter front face => This is TimeOffset value (=170 [cm]) from TS1_S04_Merlin.mcstas file.

```
Example 2:   ViewModISISver1(Face="Let_timeTest_155.mcstas", E0 = E_min, E1 = E_max,
```

modPosition=1, xw=0.196, yh = 0.12, focus_xw = 0.04, focus_yh = 0.094, dist = 0.5)

LET simulation. modPosition=1 so initial surface is at front face of shutter insert.

(IMPORTANT) If modPosition=1, the xw and yh values are obsolete. The dimensions of initial surface are automatically calculated using "RDUM" values in Let_timeTest_155.mcstas file.

In this example, focus_xw and focus_yh are arbitrary chosen to be identical to the shutter opening dimension.

N.B. Absolute normalization: The result of the Mc-Stas simulation will show neutron intensity for beam current of 1 uA.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
Face	string	Etable filename	"TS1_S04_Merlin.m
E0	meV	Lower edge of energy distribution	
E1	meV	Upper edge of energy distribution	

Name	Unit	Description	Default
modPosition	int	Defines the initial surface for neutron distribution production. Possible values = 0 (at moderator) or 1 (at shutter insert)	0
xwidth	m	Moderator width	0.12
yheight	m	Moderator height	0.12
focus_xw	m	Width of focusing rectangle	0.094
focus_yh	m	Height of focusing rectangle	0.094
dist	m	Distance from source surface to the focusing rectangle	1.7
verbose	int	Flag to output debugging information	0
beamcurrent	uA	ISIS beam current	1

Links

- Component source code found in file `ViewModISIS.comp`.

14.4. The `SNS_source` McStas Component

Modified: G. E. Granroth Nov 2014 Move to Mcstas V2 Modified: J.Y.Y. Lin April 2021 to Fix race condition Modified: P. WIllendrup 2021 Move to Move to Mcstas Version 3 and enable GPus

A source that produces a time and energy distribution from the SNS moderator files

Identification

- **Author:** G. Granroth
- **Origin:** SNS Project Oak Ridge National Laboratory
- **Date:** July 2004

Description

Produces a time-of-flight spectrum from SNS moderator files moderator files can be obtained from the SNS website. **IMPORTANT: The output units of this component are N/pulse IMPORTANT: The component needs a FULL PATH to the source input file** Notes: (1) the raw moderator files are per Sr. The focusing parameters provide the solid angle accepted by the guide to remove the Sr dependence from the output. Therefore the best practice is to set focus_xw and focus_yh to the width and height of the next beam component, respectively. The dist parameter should then be set as the distance from the moderator to the first component. (2) This component works purely by interpolation. Therefore be sure that Emin and Emax are within the limits of the moderator file

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename		Filename of source data	NULL
xwidth	m	width of moderator	0.1
yheight	m	height of moderator	0.12
dist	m	Distance from source to the focusing rectangle	
focus_xw	m	Width of focusing rectangle	
focus_yh	m	Height of focusing rectangle	
Emin	meV	minimum energy of neutron to generate	
Emax	meV	maximum energy of neutron to generate	

Links

- Component source code found in file `SNS_source.comp`.

14.5. The `SNS_source_analytic` McStas Component

A source that produces a time and energy distribution from parameterized SNS moderator files

Identification

- **Author:** F. X. Gallmeier
- **Origin:** SNS Oak Ridge National Laboratory
- **Date:** October 18, 2010

Description

Produces a time-of-flight spectrum from SNS moderator files moderator files can be obtained from the author **IMPORTANT: The output units of this component are N/pulse IMPORTANT: The component needs a FULL PATH to the source input file** Notes: (1) the raw moderator files are per Sr. The parameters focus_xw focus_yh and dist provide the solid angle with which the beam line is served. The best practice is to set focus_xw and focus_yh to the width and height of the closest beam component, and dist as the distance from the moderator location to this component. (2) Be sure that Emin and Emax are within the limits of 1e-5 to 100 eV (3) the proton pulse length T determines short- and long-pulse mode

Beamport definitions: BL11: a1Gw2-11-f5_fit_fit.dat BL14: a1Gw2-14-f5_fit_fit.dat BL17: a1Gw2-17-f5_fit_fit.dat BL02: a1Gw2-2-f5_fit_fit.dat BL05: a1Gw2-5-f5_fit_fit.dat BL08: a1Gw2-8-f5_fit_fit.dat

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename		Filename of source data	
xwidth	m	width of moderator (default 0.1 m)	0.10
yheight	m	height of moderator (default 0.12 m)	0.12
dist	m	Distance from source to the focusing rectangle (no default)	
focus_xw	m	Width of focusing rectangle (no default)	
focus_yh	m	Height of focusing rectangle (no default)	
Emin	meV	minimum energy of neutrons (default 0.01 meV)	0.01
Emax	meV	maximum energy of neutrons (default 1e5 meV)	1.0e5
tinmin	us	minimum time of neutrons (default 0 us)	0.0
tinmax	us	maximum time of neutrons (default 2000 us)	2000.0
sample_E		sample energy from: (default 0) 0="probability distribution", 1="energy range", 2="lambda range", 3="log(energy) range"	0
sample_t		sample t from: (default 0) 0="probability distribution", 1="time range"	0
proton_T	us	proton pulse length (0 us)	0.0
p_power	MW	proton power (default 2 MW)	2.0
n_pulses		number of pulses (default 1)	1.0

Links

- Component source code found in file `SNS_source_analytic.comp`.

14.6. The `Source_gen4` McStas Component

Circular/squared neutron source with flat or Maxwellian energy/wavelength spectrum (possibly spatially gaussian)

Identification

- **Author:** Emmanuel Farhi, Kim Lefmann, modified to PSI use by Jonas Okkels Birk
- **Origin:** ILL/Risoe
- **Date:** Aug 27, 2001

Description

This routine is a neutron source (rectangular or circular), which aims at a square target centered at the beam (in order to improve MC-acceptance rate). The angular divergence is then given by the dimensions of the target. The neutron energy/wavelength is distributed between Emin=E0-dE and Emax=E0+dE or Lmin=Lambda0-dLambda and Lmax=Lambda0+dLambda. The I1 may be either arbitrary (I1=0), or specified in neutrons per steradian per square cm per Å. A Maxwellian spectra may be selected if you give the source temperatures (up to 3). And a high energytail of the shape A/(lambda0-lambda) can also be specified if you give an A. Finally, a file with the flux as a function of the wavelength [lambda(Å) flux(n/s/cm^2/st/Å)] may be used with the 'flux_file' parameter. Format is 2 columns free text. Additionl distributions for the horizontal and vertical phase spaces distributions (position-divergence) may be specified with the 'xdiv_file' and 'ydiv_file' parameters. Format is free text, requiring a comment line '# xylimits: pos_min pos_max div_min div_max' to set the axis of the distribution matrix. All these files may be generated using standard monitors (better in McStas/PGPLOT format), e.g.: Monitor_nD(options="auto lambda per cm2") Monitor_nD(options="x hdiv, all auto") Monitor_nD(options="y vdiv, all auto") The source shape is defined by its radius, or can alternatively be squared if you specify nonzero h and w parameters. The beam is spatially uniform, but becomes gaussian if one of the source dimensions (radius, h or w) is negative, or you set the gaussian flag. Divergence profiles are triangular. The source may have a thickness, which will broaden the default zero time distribution. For the Maxwellian spectra, the generated intensity is dPhi/dLambda (n/s/Å) For flat spectra, the generated intensity is Phi (n/s).

Usage example: Source_gen(radius=0.1,Lambda0=2.36,dLambda=0.16, T1=20, I1=1e13) Source_gen(h=0.1,w=0.1,Emin=1,Emax=3, I1=1e13, verbose=1, gaussian=1) EXTEND %{ t = rand0max(1e-3); // set time from 0 to 1 ms for TOF instruments. %}

Some sources parameters:

```
PSI cold source T1= 296.16, I1=8.5E11, T2=40.68, I2=5.2E11
ILL VCS cold source T1=216.8,I1=1.24e+13,T2=33.9,I2=1.02e+13
(H1)         T3=16.7 ,I3=3.0423e+12
ILL HCS cold source T1=413.5,I1=10.22e12,T2=145.8,I2=3.44e13
(H5)         T3=40.1 ,I3=2.78e13
ILL Thermal tube    T1=683.7,I1=0.5874e+13,T2=257.7,I2=2.5099e+13
(H12)        T3=16.7 ,I3=1.0343e+12
ILL Hot source      T1=1695,I1=1.74e13,T2=708,I2=3.9e12
```

%VALIDATION Feb 2005: output cross-checked for 3 Maxwellians against VITESS source I(lambda), I(hor_div), I(vert_div) identical in shape and absolute values Validated by: K. Lieutenant

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
flux_file	str	Name of a two columns [lambda flux] text file that contains the wavelength distribution of the flux in [1/(s*cm**2*st*Å)]. Comments (#) and further columns are ignored. Format is compatible with McStas/PGPLOT wavelength monitor files. When specified, temperature and intensity values are ignored. NO correction for solid-angle is applied, the intensity emitted into the chosen xw x yh rectangle at distance dist.	0
xdiv_file	str	Name of the x-horiz. divergence distribution file, given as a free format text matrix, preceeded with a line '# xylimits: xmin xmax xdiv_min xdiv_max'	0
ydiv_file	str	Name of the y-vert. divergence distribution file, given as a free format text matrix, preceeded with a line '# xylimits: ymin ymax ydiv_min ydiv_max'	0
radius	m	Radius of circle in (x,y,0) plane where neutrons are generated. You may also use 'h' and 'w' for a square source	0.0
dist	m	Distance to target along z axis.	0
xw	m	Width(x) of target. If dist=0.0, xw = full horz. div in deg	0
yh	m	Height(y) of target. If dist=0.0, yh = full vert. div in deg	0
E0	meV	Mean energy of neutrons.	0
dE	meV	Energy spread of neutrons, half width.	0
Lambda0	Å	Mean wavelength of neutrons.	0
dLambda	Å	Wavelength spread of neutrons,half width	0
I1	1/(cm**2*st)	Source flux per solid angle, area and Å	0

Name	Unit	Description	Default
h	m	Source y-height, then does not use radius parameter	0
w	m	Source x-width, then does not use radius parameter	0
gaussian	0/1	Use gaussian divergence beam, normal distributions	0
verbose	0/1	display info about the source. -1 inactivate source.	0
T1	K	Temperature of the Maxwellian source, 0=none	0
flux_file_perAA	1	When true (1), indicates that flux file data is already per Angstroem. If false, file data is per wavelength bin.	0
flux_file_log	1	When true, will transform the flux table in log scale to improve the sampling. This may also cause	0
Lmin	Å	Minimum wavelength of neutrons	0
Lmax	Å	Maximum wavelength of neutrons	0
Emin	meV	Minimum energy of neutrons	0
Emax	meV	Maximum energy of neutrons	0
T2	K	Second Maxwellian source Temperature, 0=none	0
I2	1/(cm**2*st)	Second Maxwellian Source flux	0
T3	K	Third Maxwellian source Temperature, 0=none	0
I3	1/(cm**2*st)	Third Maxwellian Source flux	0
length	m	Source z-length, not anymore flat	0
phi_init	degrees	Angle abowe the x-z-plan that the source is amied at	0
theta_init	degrees	Angle from the z-axis in the x-z-plan that the source is amied at	0
HEtailA	1/(cm**2*st*Å)	Amplitude of heigh energy tail	0
HEtailL0	Å	Offset of heigh energy tail	0

Links

- Component source code found in file `Source_gen4.comp`.
- The ILL Yellow book
- P. Ageron, Nucl. Inst. Meth. A 284 (1989) 197

14.7. The `Source_multi_surfaces` McStas Component

Rectangular neutron source with subareas - using wavelength spectra reading from files

Identification

- **Author:** Ludovic Giller, Uwe Filges
- **Origin:** PSI/Villigen
- **Date:** 31.10.06

Description

This routine is a rectangular neutron source, which aims at a square target centered at the beam (in order to improve MC-acceptance rate). The angular divergence is then given by the dimensions of the target. The source surface can be divided in subareas (maximum 7 in one dimension). For each subarea a discret spectrum must be loaded given by its corresponding file. The name of the files are saved in one file and will be called with the parameter "filename". In fact the routine first reads the spectrum from files (up to 49) and it selects randomly a subarea. Secondly it generates a neutron with an random energy, selected from the spectrum corresponding to the subsurface chosen, and with a random direction of propagation within the target. ATTENTION 1: the files must be located in the working directory where also the instrument file is located ATTENTION 2: the wavelenght distribution (or binning) must be uniform

The file giving the name of sub-sources is matrix like, eg. for a 3x2 (x,y) division : spec1_1.dat spec1_2.dat spec1_3.dat spec2_1.dat spec2_2.dat spec2_3.dat(RETURN-key)

ATTENTION : The line break after the last line is important. Otherwise the files can not be read in !!!

and for example spec1_2.dat must be like, always from big to small lambda : #surface 1_2 #comments must be preceded by "#" #lambda [Å] intensity [a.u.]

```
9.0  1.39E+08
8.75  9.67E+08
8.5  1.17E+09
8.25  1.50E+09
8.0  1.60E+09
7.75  1.43E+09
7.5  1.40E+09
7.25  1.36E+09
7.0  1.35E+09
6.75  1.33E+09
6.5  1.32E+09
6.25  1.31E+09
```

[a.u.] means that your chosen unit for the intensity influences the outcoming unit. What you put in you will get out ! i.e. whatever is the unit for the input intensity you will have the same unit for the the output. Generally McStas works in neutrons per second [n/s].

Usage example: Source_multi_surfaces(yheight=0.16,xwidth=0.09,dist=1.18,xw=0.08,yh=0.15,xd
Lmin=0.01,Lmax=10.0,filename="files_name.dat")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename	str	Name of the file containing a list of the spectra file names	0
yheight	m	Source y-height	0.16
xwidth	m	Source x-width	0.09
dist	m	Distance to target along z axis	1.18
xw	m	Width(x) of target	0.08
yh	m	Height(y) of target	0.15
xdim	int	Number of subareas in the x direction	4
ydim	int	Number of subareas in the y direction	4
Emin	eV	Minimum energy of neutrons	0.0
Emax	eV	Maximum energy of neutrons	0.0
Lmin	Å	Minimum wavelength of neutrons	0.0
Lmax	Å	Maximum wavelength of neutrons	0.0

Links

- Component source code found in file `Source_multi_surfaces.comp`.

14.8. The `Source_custom` McStas Component

A flexible pulsed or continuous source with customisable parameters. Multiple pulses can be simulated over time.

Identification

- **Author:** Pablo Gila-Herranz, based on 'Source_pulsed' by K. Lieutenant
- **Origin:** Materials Physics Center (CFM-MPC), CSIC-UPV/EHU
- **Date:** May 2025

Description

Produces a custom pulse spectrum with a wavelength distribution as a sum of up to 3 Maxwellian distributions and one of undermoderated neutrons.

Usage:

By default, all Maxwellian distributions are assumed to come from anywhere in the moderator. A custom radius r_i can be defined such that the central circle is at temperature T1, and the surrounding ring is at temperature T2. The third Maxwellian distribution at temperature T3 always comes from anywhere in the moderator.

Multiple pulses can be simulated over time with the n_pulses parameter. The input flux is in units of [1/(cm^2 sr s Å)], and the output flux is per second, regardless of the number of pulses. This means that the intensity spreads over n_pulses, so that the units of neutrons per second are preserved. To simulate only the neutron counts of a single pulse, the input flux should be divided by the frequency. Bear in mind that the maximum emission time is given by t_pulse * tmax_multiplier, so short pulses with long tails may require a higher tmax_multiplier value (3 by default).

To simulate a continuous source, set a pulse length equal to the pulse period (e.g. t_pulse=1.0, freq=1.0). Note that this continuous beam is simulated over time, unlike other continuous sources in McStas which produce a Dirac delta at time 0.

Parameters xwidth and yheight must be provided for rectangular moderators, otherwise a circular shape is assumed.

Model description:

The normalised Maxwellian distribution for moderated neutrons is defined by [1] $M(\lambda)=\frac{2a^2}{T^2\lambda^5}\exp\left(-\frac{a}{T\lambda^{2}}\right)$ where $a=\left(\frac{h^2}{2m_{N}k_{B}}\right)$ and the joining function for the under-moderated neutrons is given by [1] $M(\lambda)_{um}=\frac{1}{\lambda(1+\exp(\lambda\chi-\kappa))}$

The normalised time structure of the long pulse is defined by [2] $N_{t<=t_p}=1-\exp\left(-\frac{t}{\tau/n}\right)$ $N_{t>t_p}=\exp\left(-\frac{t-t_p}{\tau}\right)-\exp\left(-\frac{t}{\tau/n}\right)$ where tp is the pulse period, tau is the pulse decay time constant, and n is the ratio of decay to ascend time constants.

Parameters for some sources:

HBS thermal source: t_pulse=0.016, freq=24.0, xwidth=0.04, yheight=0.04,

```
T1=325.0, I1=0.68e+12, tau1=0.000125,
I_um=2.47e+10, chi_um=2.5
```

HBS cold source: t_pulse=0.016, freq=24.0, radius=0.010,

```
T1=60.0, I1=1.75e+12, tau1=0.000170,
I_um=3.82e+10, chi_um=0.9
```

HBS bi-spectral:

```
t_pulse=0.016, freq=24.0, radius=0.022, r_i=0.010,
T1= 60.0, I1=1.75e+12, tau1=0.000170,
T2=305.0, I2=0.56e+12, tau2=0.000130,
I_um=3.82e+10, chi_um=2.5
```

PSI cold source: t_pulse=1, freq=1, xwidth=0.1, yheight=0.1,

```
T1=296.2, I1=8.5e+11, T2=40.68, I2=5.2e+11
```

References: [1] J. M. Carpenter et al, Nuclear Instruments and Methods in Physics Research Section A, 234, 542-551 (1985). [2] J. M. Carpenter and W. B. Yelon, Methods in Experimental Physics 23, p. 127, Chapter 2, Neutron Sources (1986).

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_index	1	Relative index of component to focus at, e.g. next is +1 (used to compute 'dist' automatically)	1
dist	m	Distance from the source to the target	0.0
focus_xw	m	Width of the target (focusing rectangle)	0.02
focus_yh	m	Height of the target (focusing rectangle)	0.02
xwidth	m	Width of the source	0.0
yheight	m	Height of the source	0.0
radius	m	Outer radius of the source (ignored if xwidth and yheight are set)	0.010
r_i	m	Radius of a central circle at temperature T1, sorrounded by a ring at temperature T2 (ignored by default)	0.0
Lmin	Å	Lower edge of the wavelength distribution	
Lmax	Å	Upper edge of the wavelength distribution	
freq	Hz	Frequency of pulses	
t_pulse	s	Proton pulse length	
tmax_multiplier	1	Defined maximum emission time at moderator (tmax = tmax_multiplier * t_pulse); 1 for continuous sources	3.0
n_pulses	1	Number of pulses to be simulated (ignored for continuous sources)	1

Name	Unit	Description	Default
T1	K	Temperature of the 1st Maxwellian distribution; for r_i > 0 it is only used for the central radii r < r_i	0.0
I1	1/(cm^2 sr s Å)	Flux per solid angle of the 1st Maxwellian distribution (integrated over the whole wavelength range)	0.0
tau1	s	Pulse decay time constant of the 1st Maxwellian distribution	0.000125
T2	K	Temperature of the 2nd Maxwellian distribution (0 to disable); for r_i > 0 it is only used for the external ring r > r_i	0.0
I2	1/(cm^2 sr s Å)	Flux per solid angle of the 2nd Maxwellian distribution	0.0
tau2	s	Pulse decay time constant of the 2nd Maxwellian distribution	0.000125
T3	K	Temperature of the 3rd Maxwellian distribution (0 to disable)	0.0
I3	1/(cm^2 sr s Å)	Flux per solid angle of the 3rd Maxwellian distribution	0.0
tau3	s	Pulse decay time constant of the 3rd Maxwellian distribution	0.000125
n_mod	1	Ratio of pulse decay constant to pulse ascend constant of moderated neutrons (n = tau_decay / tau_ascend)	1
I_um	1/(cm^2 sr s Å)	Flux per solid angle for the under-moderated neutrons	0.0
tau_um	s	Pulse decay time constant of under-moderated neutrons	0.000012
n_um	1	Ratio of pulse decay constant to pulse ascend constant of under-moderated neutrons	1
chi_um	1/Å	Factor for the wavelength dependence of under-moderated neutrons	2.5
kap_um	1	Scaling factor for the flux of under-moderated neutrons	2.2

Links

- Component source code found in file `Source_custom.comp`.
- This model is implemented in an interactive notebook to fit custom neutron sources.

14.9. The `Source_pulsed` McStas Component

A pulsed source for variable proton pulse lenghts

Identification

- **Author:** Klaus Lieutenant, based on component 'Moderator' by K. Nielsen, M. Hagen and 'ESS_moderator_long_2001' by K. Lefmann
- **Origin:** FZ Juelich
- **Date:**

Description

Produces a long pulse spectrum with a wavelength distribution as a sum of up to 3 Maxwellian distributions and one of undermoderated neutrons

It uses the time dependence of long pulses. Short pulses can, however, also be simulated by setting the proton pulse short.

If moderator width and height are given, it assumes a rectangular moderator, and otherwise a circular

Usage example: Source_pulsed(xwidth=0.04, yheight=0.04, Lmin=1.0, Lmax=3.0, t_min=0.0, t_max=0.5, dist=0.700, focus_xw=0.020, focus_yh=0.020,

```
freq=96.0, t_pulse=0.000208, T1=325.0, I1=7.6e09, tau1=0.000170, I_um=2.7e08, chi_um=
```

Parameters for some sources: HBS thermal source: xwidth=0.04, yheight=0.04, T1=325.0, I1=0.68e+12/freq, tau1=0.000125, n_mod=10, I_um=2.47e+10/freq, chi_um=2.5, t_pulse=0.016/
freq=96.0 or 24.0

```
HBS cold source   : radius=0.010,              T1= 60.0, I1=1.75e+12/freq, tau2=0.000
HBS bi-spectral   : radius=0.022, r_i=0.010,   T1= 60.0, I1=1.75e+12/freq, tau2=0.000
T2=305.0, I2=0.56e+12/freq, tau1=0.000130, n_mod= 5, I_um=3.82e+10/freq, chi_um=2.5,
```

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xwidth	m	Width of the source	0.0
yheight	m	Height of the source	0.0
radius	m	Outer radius of the source	0.010
r_i	m	Radius of a central circle that is surrounded by a ring of different temperature	0.0

Name	Unit	Description	Default
Lmin	Å	Lower edge of the wavelength distribution	
Lmax	Å	Upper edge of the wavelength distribution	
t_min	s	Lower edge of the time interval	0.0
t_max	s	Upper edge of the time interval	0.001
target_index	1	relative index of component to focus at, e.g. next is +1 this is used to compute 'dist' automatically.	1
dist	m	Distance from the source to the target	0.0
focus_xw	m	Width of the target (= focusing rectangle)	0.02
focus_yh	m	Height of the target (= focusing rectangle)	0.02
freq	Hz	Frequency of pulses	
t_pulse	s	Proton pulse length	
T1	K	Temperature of the 1st Maxwellian distribution, for r_i > 0 only for radii r in the range 0 < r < r_i	0.0
I1	1/(cm**2*sr)	Flux per solid angle of the 1st Maxwellian distribution (integrated over the whole wavelength range).	0.0
tau1	s	Pulse decay constant of the 1st Maxwellian distribution	0.000125
T2	K	Temperature of the 2nd Maxwellian distribution, 0=none, for r_i > 0 only for radii r in the range r_i < r < radius	0.0
I2	1/(cm**2*sr)	Flux per solid angle of the 2nd Maxwellian distribution	0.0
tau2	s	Pulse decay constant of the 2nd Maxwellian distribution	0.0
T3	K	Temperature of the 3rd Maxwellian distribution, 0=none	0.0
I3	1/(cm**2*sr)	Flux per solid angle of the 3rd Maxwellian distribution	0.0
tau3	s	Pulse decay constant of the 3rd Maxwellian distribution	0.0
n_mod	1	Ratio of pulse decay constant to pulse ascend constant of moderated neutrons	10
I_um	1/(cm**2*sr)	Flux per solid angle for the undermoderated neutrons	0.0

Name	Unit	Description	Default
tau_um	s	Pulse decay constant of under-moderated neutrons	0.000012
n_um	1	Ratio of pulse decay constant to pulse ascend constant of under-moderated neutrons	5
chi_um	1/Å	Factor for the wavelength dependence of under-moderated neutrons	2.5
kap_um	1	Scaling factor for the flux of under-moderated neutrons	2.2

Links

- Component source code found in file `Source_pulsed.comp`.

14.10. Contributed guides and benders

14.11. The `Guide_curved` McStas Component

Non-focusing curved neutron guide.

Identification

- **Author:** Ross Stewart
- **Origin:** ILL (France).
- **Date:** November 18 2003

Description

Models a rectangular curved guide tube with entrance centered on the Z axis. The entrance lies in the X-Y plane. Draws a true depiction of the guide, and trajectories. Guide is not focusing.

```
Example: Guide_curved(w1=0.1, h1=0.1, l=2.0, R0=0.99, Qc=0.021,
alpha=6.07, m=2, W=0.003, curvature=2700)
```

%BUGS This component does not work with gravitation on. Use component Guide_gravity then. Systematic error on transmitted flux is found to be about 10%.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
w1	m	Width at the guide entry	
h1	m	Height at the guide entry	
l	m	length of guide	
R0	1	Low-angle reflectivity	0.995
Qc	$Å^{-1}$	Critical scattering vector	0.0218
alpha	Å	Slope of reflectivity	4.38
m	1	m-value of material. Zero means completely absorbing.	2
W	$Å^{-1}$	Width of supermirror cut-off	0.003
curvature	m	Radius of curvature of the guide	2700

Links

- Component source code found in file `Guide_curved.comp`.
- Bender

14.12. The `Guide_four_side` McStas Component

Guide with four side walls

Identification

- **Author:** Tobias Panzner
- **Origin:** PSI
- **Date:** 07/08/2010

Description

This component models a guide with four side walls. As user you can controll the properties of every wall separatly. All togther you have up to 8 walls: 4 inner walls and 4 outer walls.

Every single wall can have a elliptic, parabolic or straight shape. All four sides of the guide are independent from each other. In the elliptic case the side wall shape follows the equation x^2/b^2+(z+z0)^2/a^2=1 (the center of the ellipse is located at (0,-z0)). In the parabolic case the side wall shape follows the equation z=b-ax^2;mc In the straight case the side wall shape follows the equation z=l/(w2-w1)*x-w1.

The shape selection is done by the focal points. The focal points are located at the z-axis and are defined by their distance to the entrance or exit window of the guide (in the following called 'focal length').

If both focal lengths for one wall are zero it will be a straight wall (entrance and exit width have to be given in the beginning).

If one of the focal lengths is not zero the shape will be parabolic (only the entrance width given in the beginning is recognized; exit width will be calculated). If the the entrance focal length is zero the guide will be a focusing devise. If the exit focal length is zero it will be defocusing devise.

If both focals are non zero the shape of the wall will be elliptic (only the entrance width given in the beginning is recognized; exit width will be calculated).

Notice: 1.)The focal points are in general located outside the guide (positive focal lengths). Focal points inside the guide need to have negative focal lengths. 2.)The exit width parameters (w2r, w2l, h2u,h2d) are only taken into account if the walls have a linear shape. In the ellitic or parabolic case they will be ignored.

For the inner channel: the outer side of each wall is calculated by the component in depentence of the wallthickness and the shape of the inner side.

Each of walls can have a own indepenting reflecting layer (defined by an input file) or it can be a absorber or it can be transparent.

The reflectivity properties can be given by an input file (Format [q(Å-1) R(0-1)]) or by parameters (Qc, alpha, m, W).

%BUGS This component does not work with gravitation on.

This component does not work correctly in GROUP-modus.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
RIreflect		(str) Name of relfectivity file for the right inner wall.	0
LIreflect		(str) Name of relfectivity file for the left inner wall.	0
UIreflect		(str) Name of relfectivity file for the top inner wall.	0
DIreflect		(str) Name of relfectivity file for the bottom inner wall.	0
ROreflect		(str) Name of relfectivity file for the right outer wall.	0
LOreflect		(str) Name of relfectivity file for the left outer wall.	0
UOreflect		(str) Name of relfectivity file for the top outer wall.	0

Name	Unit	Description	Default
DOreflect		(str) Name of relfectivity file for the bottom outer wall.	0
w1l	m	Width at the left guide entry (positive x-axis) (DEFAULT = 0)	0.002
w2l	m	Width at the left guide exit (positive x-axis) (DEFAULT = 0)	0.002
linwl	m	left horizontal wall: distance of 1st focal point and guide entry (DEFAULT = 0)	0
loutwl	m	left horizontal wall: distance of 2nd focal point and guide exit (DEFAULT = 0)	0
w1r	m	Width at the right guide entry (negative x-axis) (DEFAULT = 0)	0.002
w2r	m	Width at the right guide exit (negative x-axis) (DEFAULT = 0)	0.002
linwr	m	right horizontal wall: distance of 1st focal point and guide entry (DEFAULT = 0)	0.0
loutwr	m	right horizontal wall: distance of 2nd focal point and guide exit (DEFAULT = 0)	0
h1u	m	Height at the top guide entry (positive y-axis) (DEFAULT = 0)	0.002
h2u	m	Height at the top guide entry (positive y-axis) (DEFAULT = 0)	0.002
linhu	m	upper vertical wall: distance of 1st focal point and guide entry (DEFAULT = 0)	0.0
louthu	m	upper vertical wall: distance of 2nd focal point and guide exit (DEFAULT = 0)	0
h1d	m	Height at the bottom guide entry (negative y-axis) (DEFAULT = 0)	0.002
h2d	m	Height at the bottom guide entry (negative y-axis) (DEFAULT = 0)	0.002
linhd	m	lower vertical wall: distance of 1st focal point and guide entry (DEFAULT = 0)	0.0
louthd	m	lower vertical wall: distance of 2nd focal point and guide exit (DEFAULT = 0)	0
l	m	length of guide (DEFAULT = 0)	0
R0	1	Low-angle reflectivity (DEFAULT = 0.99)	0.99
Qcxl	$Å^{-1}$	Critical scattering vector for left vertical inner wall (DEFAULT = 0.0217)	0.0217

Name	Unit	Description	Default
Qcxr	$Å^{-1}$	Critical scattering vector for right vertical inner wall (DEFAULT = 0.0217)	0.0217
Qcyu	$Å^{-1}$	Critical scattering vector for top inner wall (DEFAULT = 0.0217)	0.0217
Qcyd	$Å^{-1}$	Critical scattering vector for bottom inner wall (DEFAULT = 0.0217)	0.0217
alphaxl	Å	Slope of reflectivity for left vertical inner wall (DEFAULT = 6.07)	6.07
alphaxr	Å	Slope of reflectivity for right vertical inner wall (DEFAULT = 6.07)	6.07
alphayu	Å	Slope of reflectivity for top inner wall (DEFAULT = 6.07)	6.07
alphayd	Å	Slope of reflectivity for bottom inner wall (DEFAULT = 6.07)	6.07
Wxr	$Å^{-1}$	Width of supermirror cut-off for right inner wall (DEFAULT = 0.003)	0.003
Wxl	$Å^{-1}$	Width of supermirror cut-off for left inner wall (DEFAULT = 0.003)	0.003
Wyu	$Å^{-1}$	Width of supermirror cut-off for top inner wall (DEFAULT = 0.003)	0.003
Wyd	$Å^{-1}$	Width of supermirror cut-off for bottom inner wall (DEFAULT = 0.003)	0.003
mxr	1	m-value of material for right vertical inner wall. 0 means the wall is absorbing. -1 means the wall is transparent. (DEFAULT = 3.6)	3.6
mxl	1	m-value of material for left vertical inner wall. 0 means the wall is absorbing. -1 means the wall is transparent. (DEFAULT = 3.6)	3.6
myu	1	m-value of material for top inner wall 0 means the wall is absorbing. -1 means the wall is transparent. (DEFAULT = 3.6)	3.6
myd	1	m-value of material for bottom inner wall 0 means the wall is absorbing. -1 means the wall is transparent. (DEFAULT = 3.6)	3.6
QcxrOW	$Å^{-1}$	Critical scattering vector for right vertical outer wall (DEFAULT = 0.0217)	0.0217
QcxlOW	$Å^{-1}$	Critical scattering vector for left vertical outer wall (DEFAULT = 0.0217)	0.0217

Name	Unit	Description	Default
QcyuOW	$Å^{-1}$	Critical scattering vector for top outer wall (DEFAULT = 0.0217)	0.0217
QcydOW	$Å^{-1}$	Critical scattering vector for bottom outer wall (DEFAULT = 0.0217)	0.0217
alphaxlOW	Å	Slope of reflectivity for left vertical outer wall (DEFAULT = 6.07)	6.07
alphaxrOW	Å	Slope of reflectivity for right vertical outer wall (DEFAULT = 6.07)	6.07
alphayuOW	Å	Slope of reflectivity for top outer wall (DEFAULT = 6.07)	6.07
alphaydOW	Å	Slope of reflectivity for bottom outer wall (DEFAULT = 6.07)	6.07
WxrOW	$Å^{-1}$	Width of supermirror cut-off for right outer wall (DEFAULT = 0.003)	0.003
WxlOW	$Å^{-1}$	Width of supermirror cut-off for left outer wall (DEFAULT = 0.003)	0.003
WyuOW	$Å^{-1}$	Width of supermirror cut-off for top outer wall (DEFAULT = 0.003)	0.003
WydOW	$Å^{-1}$	Width of supermirror cut-off for bottom outer wall (DEFAULT = 0.003)	0.003
mxrOW	1	m-value of material for right vertical outer wall 0 means the wall is absorbing. (DEFAULT) -1 means the wall is transparent. (DEFAULT = 0)	0
mxlOW	1	m-value of material for left vertical outer wall 0 means the wall is absorbing.(DEFAULT) -1 means the wall is transparent. (DEFAULT = 0)	0
myuOW	1	m-value of material for top outer wall 0 means the wall is absorbing. (DEFAULT) -1 means the wall is transparent. (DEFAULT = 0)	0
mydOW	1	m-value of material for bottom outer wall 0 means the wall is absorbing. (DEFAULT) -1 means the wall is transparent. (DEFAULT = 0)	0
rwallthick	m	thickness of the right wall (DEFAULT = 0.001 m)	0.001
lwallthick	m	thickness of the left wall (DEFAULT = 0.001 m)	0.001
uwallthick	m	thickness of the top wall (DEFAULT = 0.001 m)	0.001

Name	Unit	Description	Default
dwallthick	m	thickness of the bottom wall(DEFAULT = 0.001 m)	0.001

Links

- Component source code found in file `Guide_four_side.comp`.

14.13. The `Guide_gravity_psd` McStas Component

Neutron straight guide with gravity. Can be channeled and focusing. Waviness may be specified, as well as side chamfers (on substrate).

Identification

- **Author:** Emmanuel Farhi
- **Origin:** ILL (France).
- **Date:** Aug 03 2001

Description

Models a rectangular straight guide tube centered on the Z axis, with gravitation handling. The entrance lies in the X-Y plane. The guide can be channeled (nslit,d,nhslit parameters). The guide coating specifications may be entered via different ways (global, or for each wall m-value).

Waviness (random) may be specified either globally or for each mirror type. Side chamfers (due to substrate processing) may be specified the same way. In order to model a realistic straight guide assembly, a long guide of length 'l' may be splitted into 'nelements' between which chamfers/gaps are positioned.

The reflectivity may be specified either using an analytical mode (see Component manual) or as a text file in free format, with 1st column as q[Å-1] and 2nd column as the reflectivity R in [0-1]. For details on the geometry calculation see the description in the McStas component manual.

There is a special rotating mode in order to approximate a Fermi Chopper behaviour, in the case the neutron trajectory is nearly linear inside the chopper slits, i.e. the neutrons are fast w/r/ to the chopper speed. Slits are straight, but may be super-mirror coated. In this case, the component is NOT centered, but located at its entry window. It should then be shifted by -l/2.

Example: Guide_gravity_psd(w1=0.1, h1=0.1, w2=0.1, h2=0.1, l=12,

```
R0=0.99, Qc=0.0219, alpha=6.07, m=1.0, W=0.003)
```

%VALIDATION May 2005: extensive internal test, all problems solved Validated by: K. Lieutenant

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nxpsd	1	Number of bins in the built-in mirror parallel monitors	
filenameT	str	Filename for top-mirror-monitor	
filenameB	str	Filename for bottom-mirror-monitor	
filenameL	str	Filename for left-mirror-monitor	
filenameR	str	Filename for right-mirror-monitor	
w1	m	Width at the guide entry	
h1	m	Height at the guide entry	
w2	m	Width at the guide exit. If 0, use w1.	0
h2	m	Height at the guide exit. If 0, use h1.	0
l	m	length of guide	
R0	1	Low-angle reflectivity	0.995
Qc	$Å^{-1}$	Critical scattering vector	0.0218
alpha	Å	Slope of reflectivity	4.38
m	1	m-value of material. Zero means completely absorbing. m=0.65 glass/SiO2 Si Ni Ni58 supermirror Be Diamond m= 0.65 0.47 1 1.18 2-6 1.01 1.12 for glass/SiO2, m=1 for Ni, 1.2 for Ni58, m=2-6 for supermirror. m=0.47 for Si	1.0
W	$Å^{-1}$	Width of supermirror cut-off	0.003
nslit	1	Number of vertical channels in the guide (>= 1) (nslit-1 vertical dividing walls).	1
d	m	Thickness of subdividing walls	0.0005
mleft	1	m-value of material for left. vert. mirror	-1
mright	1	m-value of material for right. vert. mirror	-1
mtop	1	m-value of material for top. horz. mirror	-1
mbottom	1	m-value of material for bottom. horz. mirror	-1
nhslit	1	Number of horizontal channels in the guide (>= 1). (nhslit-1 horizontal dividing walls). this enables to have nslit*nhslit rectangular channels	1

Name	Unit	Description	Default
G	m/s^2	Gravitation norm. 0 value disables G effects.	0
aleft	1	alpha-value of left vert. mirror	-1
aright	1	alpha-value of right vert. mirror	-1
atop	1	alpha-value of top horz. mirror	-1
abottom	1	alpha-value of left horz. mirror	-1
wavy	deg	Global guide waviness	0
wavy_z	deg	Partial waviness along propagation axis	0
wavy_tb	deg	Partial waviness in transverse direction for top/bottom mirrors	0
wavy_lr	deg	Partial waviness in transverse direction for left/right mirrors	0
chamfers	m	Global chamfers specifications (in/out/mirror sides).	0
chamfers_z	m	Input and output chamfers	0
chamfers_lr	m	Chamfers on left/right mirror sides	0
chamfers_tb	m	Chamfers on top/bottom mirror sides	0
nelements	1	Number of sections in the guide (length l/nelements).	1
nu	Hz	Rotation frequency (round/s) for Fermi Chopper approximation	0
phase	deg	Phase shift for the Fermi Chopper approximation	0
reflect	str	Reflectivity file name. Format q(Å-1) R(0-1)	"NULL"

Links

- Component source code found in file `Guide_gravity_psd.comp`.

14.14. The `Guide_honeycomb` McStas Component

Neutron guide with gravity and honeycomb geometry. Can be channeled and focusing.

Identification

- **Author:** G. Venturi
- **Origin:** ILL (France).
- **Date:** Apr 2003

Description

Models a honeycomb guide tube centered on the Z axis. The entrance lies in the X-Y plane. Gravitation applies also when reaching the guide input window. The guide can be channeled (hexagonal channel section; nslit,d parameters). The guide coating specifications may be entered via different ways (global, or for each wall m-value). For details on the geometry calculation see the description in the McStas reference manual.

Example: Guide_honeycomb(w1=0.1, w2=0.1, l=12,

```
R0=0.99, Qc=0.0219, alpha=6.07, m=1.0, W=0.003, nslit=1, d=0.0005)
```

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
w1	m	Width at the guide entry	
w2	m	Width at the guide exit. If zero, sets w2=w1.	0
l	m	length of guide	
R0	1	Low-angle reflectivity	0.995
Qc	$Å^{-1}$	Critical scattering vector	0.0218
alpha	Å	Slope of reflectivity	4.38
m	1	m-value of material. Zero means completely absorbing.	1.0
W	$Å^{-1}$	Width of supermirror cut-off	0.003
nslit	1	Number of horizontal channels in the guide (>= 1) [1] (nslit-1 vertical dividing walls)	1
d	m	Thickness of subdividing walls [0]	0.0005
mleftup	1	m-value of material for leftup. oblique mirror	-1
mrightup	1	m-value of material for rightdown. oblique mirror	-1
mleftdown	1	m-value of material for rightdown. oblique mirror	-1
mrightdown	1	m-value of material for leftup. oblique mirror	-1
mleft	1	m-value of material for left. vertical mirror	-1
mright	1	m-value of material for right. vertical mirror	-1

Name	Unit	Description	Default
G	m/s^2	Gravitation norm. 0 value disables G effects.	0

Links

- Component source code found in file `Guide_honeycomb.comp`.

14.15. The `Guide_m` McStas Component

Release: McStas 1.12c
Neutron guide.

Identification

- **Author:** Kristian Nielsen
- **Origin:** Risoe
- **Date:** September 2 1998, Modified 2013

Description

Models a rectangular guide tube centered on the Z axis. The entrance lies in the X-Y plane. For details on the geometry calculation see the description in the McStas reference manual. The reflectivity profile may either use an analytical mode (see Component Manual) or a 2-columns reflectivity free text file with format

```
[q(Angs-1) R(0-1)].
```

```
Example: Guide(w1=0.1, h1=0.1, w2=0.1, h2=0.1, l=2.0,
R0=0.99, Qc=0.021, alpha=6.07, m=2, W=0.003
```

%VALIDATION May 2005: extensive internal test, no bugs found Validated by: K. Lieutenant
%BUGS This component does not work with gravitation on. Use component Guide_gravity then.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
reflect	str	Reflectivity file name. Format [q(Å^-1) R(0-1)]	0
w1	m	Width at the guide entry	
h1	m	Height at the guide entry	
w2	m	Width at the guide exit	
h2	m	Height at the guide exit	
l	m	Length of guide	
R0_left	1	Low-angle reflectivity, left mirrror.	0.99
R0_right	1	Low-angle reflectivity, right mirror.	0.99
R0_top	1	Low-angle reflectivity, top mirror.	0.99
R0_bottom	1	Low-angle reflectivity, bottom mirror.	0.99
Qc_left	$Å^{-1}$	Critical scattering vector, left mirrror.	0.0219
Qc_right	$Å^{-1}$	Critical scattering vector, right mirror.	0.0219
Qc_top	$Å^{-1}$	Critical scattering vector, top mirror.	0.0219
Qc_bottom	$Å^{-1}$	Critical scattering vector, bottom mirror.	0.0219
aleft	Å	Slope of reflectivity, left mirrror.	6.07
aright	Å	Slope of reflectivity, right mirror.	6.07
atop	Å	Slope of reflectivity, top mirror.	6.07
abottom	Å	Slope of reflectivity, bottom mirror.	6.07
mleft	1	m-value of material. Zero means completely absorbing, left mirrror.	2
mright	1	m-value of material. Zero means completely absorbing, right mirror.	2
mtop	1	m-value of material. Zero means completely absorbing, top mirror.	2
mbottom	1	m-value of material. Zero means completely absorbing, bottom mirror.	2
W_left	$Å^{-1}$	Width of supermirror cut-off, left mirror.	0.003
W_right	$Å^{-1}$	Width of supermirror cut-off, right mirror.	0.003
W_top	$Å^{-1}$	Width of supermirror cut-off, top mirror.	0.003
W_bottom	$Å^{-1}$	Width of supermirror cut-off, bottom mirror.	0.003
W	$Å^{-1}$	Overall width of supermirror cut-off, overwrites values from individual mirrors	0
m	1	Overall m-value of supermirrors, overwrites values from individual mirrors	0
alpha	Å	Overall slope of reflectivity, overwrites values from individual mirrors	0
R0	1	Overall, low-angle reflectivity, overwrites values from individual mirrors	0

Name	Unit	Description	Default
Qc	$Å^{-1}$	Overall, critical scattering vector, overwrites values from individual mirrors	0

Links

- Component source code found in file `Guide_m.comp`.

14.16. The `Guide_multichannel` McStas Component

Multichannel neutron guide with semi-transparent blades. Derived from Guide_channeled by Christian Nielsen. Allows to simulate bi-spectral extraction optics.

Identification

- **Author:** Jan Saroun (saroun@ujf.cas.cz)
- **Origin:** Nuclear Physics Institute, CAS, Rez
- **Date:** 17.3.2022

Description

Models a rectangular guide with equidistant vertical blades of finite thickness. The blades material can be either fully absorbing or semi-transparent. The absorption coefficient is wavelength dependent according to the semi-empirical model used e.g. in J. Baker et al., J. Appl. Cryst. 41 (2008) 1003 or A. Freund, Nucl. Instr. Meth. A 213 (1983) 495. Data are provided for Si and Al2O3.

All walls are flat, curvature is not implemented (may be added as a future upgrade) Tapering is possible by setting different entry ad exit dimensions. Different guide coating can be set for vertical and horizontal mirrors. For transparent walls, neutrons are alloed to migrate between channels and to propagate through the blades.

The model is almost equivalent to the GUIDE component in SIMRES (http://neutron.ujf.cas.cz/rest when used with zero curvature and type set to "guide or bender". The features from SIMRES not included in this McSas model are: - has a more conservative model for absorption in blades: events above r(m<mc) are automatically ABSORBED. - defines waviness - works with bent geometry - works with gravity - allows for 2D grid of blades

bug fix 24/3/2017: incorrect handling of transition into blades = no transmission

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
w1	m	Width at the guide entry	
h1	m	Height at the guide entry	
w2	m	Width at the guide exit	0
h2	m	Height at the guide exit	0
l	m	Length of guide	
R0	1	Low-angle reflectivity	0.995
Qc	$Å^{-1}$	Critical scattering vector	0
alpha	Å	Slope of reflectivity	0
m	1	m-value of material. Zero means completely absorbing.	0
nslit	1	Number of channels in the guide (>= 1)	1
dlam	m	Thickness of lamellae	0.0005
Qcx	$Å^{-1}$	Critical scattering vector for left and right vertical mirrors in each channel	0.0218
Qcy	$Å^{-1}$	Critical scattering vector for top and bottom mirrors	0.0218
alphax	Å	Slope of reflectivity for left and right vertical mirrors in each channel	4.38
alphay	Å	Slope of reflectivity for top and bottom mirrors	4.38
W	$Å^{-1}$	Width of supermirror cut-off for all mirrors	0.003
mx	1	m-value of material for left and right vertical mirrors in each channel. Zero means completely absorbing.	1
my	1	m-value of material for top and bottom mirrors. Zero means completely absorbing.	1
mater	string	"Si", "Al2O3", or "absorb" (default)	"absorb"

Links

- Component source code found in file `Guide_multichannel.comp`.

14.17. The `Vertical_Bender` McStas Component

Release: McStas 2.4

Multi-channel bender curving vertically down.

Identification

- **Author:** Andrew Jackson, Richard Heenan

- **Origin:** ESS
- **Date:** August 2016, June 2017

Description

Based on Pol_bender written by Peter Christiansen Models a rectangular curved guide with entrance on Z axis. Entrance is on the X-Y plane. Draws a correct depiction of the guide with multiple channels - i.e. following components need to be displaced. Guide can contain multiple channels using horizontal blades Reflectivity modeled using StdReflecFunc and {R0, Qc, alpha, m, W} can be set for top (outside of curve), bottom (inside of curve) and sides (both sides equal), blades reflectivities match top and bottom (each channel is like a "mini guide"). Neutrons are tracked, with gravity, inside the wall of a cylinder using distance steps of diststep1. Upon crossing the inner or outer wall, the final step is repeated using steps of diststep2, thus checking more closely for the first crossing of either wall. If diststep1 is too large than a "grazing incidence reflection" may be missed altogether. If diststep1 is too small, then the code will run slower! Exact intersection tmes with the flat sides and ends of the bender channel are calculated first in order to limit the time spent tracking curved trajectories. Turning gravity off will not make this run any faster, as the same method is used.

28/06/2017 still need validation against other codes (e.g. for gravity off case)

Example: Vertical_Bender(xwidth = 0.05, yheight = 0.05, length = 3.0, radius = 70.0, nslit = 5, d=0.0005, diststep1=0.020, diststep2=0.002,

```
rTopPar={0.99, 0.219, 6.07, 3.0, 0.003},
rBottomPar={0.99, 0.219, 6.07, 2.0, 0.003},
rSidesPar={0.99, 0.219, 6.07, 2.0, 0.003},
```

See example instrument Test_Vert_Bender

%BUGS Original code did not work with rotation about axes and gravity. This tedious tracking method should work (28/6/17 still under test) for a vertical bender, with optional rotation about x axis and gravity (and likely rotation about y axis but needs testing, and even perhaps rotation about z axis as in rotated local frame Gx then is non zero but the edge solver still works). Would also need test the drawing in trace mode).

GRAVITY : Yes, when component is not rotated

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
rTopPar	vector	Parameters for reflectivity of bender top surface	{0.99, 0.219, 6.07, 0.0, 0.003}

Name	Unit	Description	Default
rBottomPar	vector	Parameters for reflectivity of bender bottom surface	{0.99, 0.219, 6.07, 0.0, 0.003}
rSidesPar	vector	Parameters for reflectivity of bender sides surface	{0.99, 0.219, 6.07, 0.0, 0.003}
xwidth	m	Width at the guide entry	
yheight	m	Height at the guide entry	
length	m	length of guide along center	
radius	m	Radius of curvature of the guide (+:curve up/-:curve down)	
G	m/s^2	Gravitational acceleration (currently unused)	9.8
nchan	1	Number of channels	1
d	m	Width of spacers (subdividing absorbing walls)	0.0
debug	1	0 to 10 for zero to maximum print out - reduce number of neutrons to run !	0
endFlat	1	If endflat>0 then entrance and exit planes are parallel.	0
drawOption	1	Drawing detail for mcdisplay, 1: fine, 2: normal, 3: rough	1
alwaystrack	1	default 0, 1 to force tracking even when gravity is off	0
diststep1	m	inital collision search steps for trajectory in cylinder when gravity is non zero	0.020
diststep2	m	final steps for trajectory in cylinder	0.002

Links

- Component source code found in file `Vertical_Bender.comp`.

14.18. The `NMO` McStas Component

Date: 2022-2023
NMO (nested mirror optic) modules

Identification

- **Author:** Christoph Herb, TUM
- **Origin:** TUM

- **Date:** 2022-2023

Description

Simulates NMO (nested mirror optic) modules as concevied by Böni et al., see Christoph Herb et al., Nucl. Instrum. Meth. A 1040, 1671564 (1-18) 2022.

The component relies on an updated version of conic.h from MIT.

NMO.comp contains no actual code, uses INHERIT of all sections from FlatEllipse_finite_mirror

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sourceDist	m	Distance used for calculating the spacing of the mirrors, 0: use LStart	0
LStart	m	Left focal point	0.6
LEnd	m	Right focal point	0.6
lStart	m	z-Value of the mirror start	0.0
lEnd	m	z-Value of the mirror end	0.0
r_0	m	distance to the mirror at lStart	0.02076
nummirror	1	number of mirrors	9
mf		mvalue of the inner side of the coating, m>10 results in perfect reflections	4
mb		mvalue of the outer side of the coating, m>10 results in perfect reflections	0
R0	1	Low-angle reflectivity	0.99
Qc	$Å^{-1}$	critical scattering vector	0.021
W	$Å^{-1}$	width of supermirror cutoff	0.003
alpha	Å	Slope of reflectivity	6.07
mirror_width	m	width of the individual mirrors	0.003
mirror_sidelength	m	side length of the individual mirrors	1
doubleReflections	1	binary value determining whether the mirror backside is reflective	0
rfront_inner_file	str	file of distances to the optical axis of the individual mirrors	"NULL"

Links

- Component source code found in file `NMO.comp`.
- Christoph Herb et al., Nucl. Instrum. Meth. A 1040, 1671564 (1-18) 2022.

14.19. The `Conics_EH` McStas Component

Release: McStas 2.7 Date: September 2021
Elliptic-Hyperbolic shells for Wolter optic

Identification

- **Author:** Peter Wilendrup and Erik Knudsen
(derived from Giacomo Resta skeleton-component)
- **Origin:** DTU
- **Date:** September 2021

Description

Implements a set of nshells Wolter Ellipsoid/Hyperboloid pairs using conics.h from ConicTracer.

The component has two distinct modes of specifying the geometry: a) Via the radii vector, parametrized from largest to smallest radius with a length of 'nshells'

b) By specifying radii rmax and rmin, between which a quadratic law distributes 'nshells' surfaces.

The mirrors are assumed to be touching at the mid-optic plane, i.e. there is no gap between primary and secondary mirror. By definition the ratio between primary and secondary mirror glancing angles is 1/3. At present a single m-value is used for all mirrors.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
rmin	m	Midoptic plane radius of innermost mirror pair.	0.0031416
rmax	m	Midoptic plane radius of outermost mirror pair.	0.05236
focal_length_u	m	Focal length (upstream) of the mirror pairs.	10
focal_length_d	m	Focal length (downstream) of the mirror pairs.	10
le	m	Paraboloid mirror length.	0.25
lh	m	Hyperboloid mirror length.	0.25
nshells	1	Number of nested shells to expect	4
m	1	Critical angle of mirrors as multiples of Ni_c.	1

Name	Unit	Description	Default
mirr_thick	m	Thickness of mirror shell surfaces - NOT YET IMPLEMENTED	0
disk		Flag. If nonzero, insert a disk to block the central area within the innermost mirror.	1
radii	m	Optional vector of radii (length should match nshells)	NULL
R0	1	Reflectivity at Low angles for reflectivity curve approximation	0.99
Qc	$Å^{-1}$	Critical scattering vector	0.021
W	$Å^{-1}$	Width of supermirror cut-off	0.003
alpha	Å	Slope of reflectivity for reflectivity curve approximation	6.07
transmit	1	Fraction of statistics to assign to transmitted beam - NOT YET IMPLEMENTED	0
sigma_m	rad	RMS meridional slope (figure) error of the mirror surfaces; for grazing angles below 10*sigma_m the waviness model of Hansen et al., arXiv:1612.03763 is used	1e-6
sigma_s	rad	RMS sagittal slope (figure) error of the mirror surfaces	250e-6

Links

- Component source code found in file `Conics_EH.comp`.

14.20. The `Conics_HE` McStas Component

Release: McStas 2.7 Date: September 2021
Hyperbolic-Elliptic shells for Wolter optic

Identification

- **Author:** Peter Wilendrup and Erik Knudsen
(derived from Giacomo Resta skeleton-component)
- **Origin:** DTU
- **Date:** September 2021

Description

Implements a set of nshells Wolter Ellipsoid/Hyperboloid pairs using conics.h from ConicTracer.

The component has two distinct modes of specifying the geometry: a) Via the radii vector, parametrized from largest to smallest radius with a length of 'nshells'

b) By specifying radii rmax and rmin, between which a quadratic law distributes 'nshells' surfaces.

The mirrors are assumed to be touching at the mid-optic plane, i.e. there is no gap between primary and secondary mirror. By definition the ratio between primary and secondary mirror glancing angles is 1/3. At present a single m-value is used for all mirrors.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
rmin	m	Midoptic plane radius of innermost mirror pair.	0.0031416
rmax	m	Midoptic plane radius of outermost mirror pair.	0.05236
focal_length_u	m	Focal length (upstream) of the mirror pairs.	10
focal_length_d	m	Focal length (downstream) of the mirror pairs.	10
le	m	Paraboloid mirror length.	0.25
lh	m	Hyperboloid mirror length.	0.25
nshells	1	Number of nested shells to expect	4
m	1	Critical angle of mirrors as multiples of Ni_c.	1
mirr_thick	m	Thickness of mirror shell surfaces - NOT YET IMPLEMENTED	0
disk		Flag. If nonzero, insert a disk to block the central area within the innermost mirror.	1
radii	m	Optional vector of radii (length should match nshells)	NULL
R0	1	Reflectivity at Low angles for reflectivity curve approximation	0.99
Qc	$Å^{-1}$	Critical scattering vector	0.021
W	$Å^{-1}$	Width of supermirror cut-off	0.003
alpha	Å	Slope of reflectivity for reflectivity curve approximation	6.07

Name	Unit	Description	Default
transmit	1	Fraction of statistics to assign to transmitted beam - NOT YET IMPLEMENTED	0
sigma_m	rad	RMS meridional slope (figure) error of the mirror surfaces; for grazing angles below 10*sigma_m the waviness model of Hansen et al., arXiv:1612.03763 is used	1e-6
sigma_s	rad	RMS sagittal slope (figure) error of the mirror surfaces	250e-6

Links

- Component source code found in file `Conics_HE.comp`.

14.21. The `Conics_PH` McStas Component

Release: McStas 2.7 Date: September 2021
Parabolid-Hyperbolic shells for Wolter optic

Identification

- **Author:** Peter Wilendrup and Erik Knudsen
(derived from Giacomo Resta skeleton-component)
- **Origin:** DTU
- **Date:** September 2021

Description

Implements a set of nshells Wolter I Paraboloid/Hyperboloid pairs using conics.h from ConicTracer.

The component has two distinct modes of specifying the geometry: a) Via the radii vector, parametrized from largest to smallest radius with a length of 'nshells'

b) By specifying radii rmax and rmin, between which a quadratic law distributes 'nshells' surfaces.

The mirrors are assumed to be touching at the mid-optic plane, i.e. there is no gap between primary and secondary mirror.* By definition the ratio between primary and secondary mirror glancing angles is 1/3. At present a single m-value is used for all mirrors.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
rmin	m	Midoptic plane radius of innermost mirror pair.	0.325
rmax	m	Midoptic plane radius of outermost mirror pair.	0.615
focal_length	m	Focal length of the mirror pairs.	10.070
lp	m	Paraboloid mirror length.	0.84
lh	m	Hyperboloid mirror length.	0.84
nshells	1	Number of nested shells to expect	4
m	1	Critical angle of mirrors as multiples of Ni.	1
mirr_thick	m	Thickness of mirror shell surfaces - NOT YET IMPLEMENTED	0
disk		Flag. If nonzero, insert a disk to block the central area within the innermost mirror.	1
radii	m	Optional vector of radii (length should match nshells)	NULL
R0	1	Reflectivity at Low angles for reflectivity curve approximation	0.99
Qc	$Å^{-1}$	Critical scattering vector	0.021
W	$Å^{-1}$	Width of supermirror cut-off	0.003
alpha	Å	Slope of reflectivity for reflectivity curve approximation	6.07
transmit	1	Fraction of statistics to assign to transmitted beam - NOT YET IMPLEMENTED	0
sigma_m	rad	RMS meridional slope (figure) error of the mirror surfaces; for grazing angles below 10*sigma_m the waviness model of Hansen et al., arXiv:1612.03763 is used	1e-6
sigma_s	rad	RMS sagittal slope (figure) error of the mirror surfaces	250e-6

Links

- Component source code found in file `Conics_PH.comp`.

14.22. The `Conics_PP` McStas Component

Release: McStas 2.7 Date: September 2021
Parabolid-Parabolid shells for Wolter optic

Identification

- **Author:** Peter Wilendrup and Erik Knudsen
(derived from Giacomo Resta skeleton-component)
- **Origin:** DTU
- **Date:** September 2021

Description

Implements a set of nshells Wolter I Paraboloid/Parabolid pairs using conics.h from ConicTracer.

The component has two distinct modes of specifying the geometry: a) Via the radii vector, parametrized from largest to smallest radius with a length of 'nshells'

b) By specifying radii rmax and rmin, between which a quadratic law distributes 'nshells' surfaces.

The mirrors are assumed to be touching at the mid-optic plane, i.e. there is no gap between primary and secondary mirror. By definition the ratio between primary and secondary mirror glancing angles is 1/3. At present a single m-value is used for all mirrors.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
rmin	m	Midoptic plane radius of innermost mirror pair.	0.0031416
rmax	m	Midoptic plane radius of outermost mirror pair.	0.05236
focal_length_u	m	Focal length of upstream mirror.	10.070
focal_length_d	m	Focal length of downstream mirror.	10.070
lp1	m	Upstream paraboloid mirror length.	0.84
lp2	m	Downstream paraboloid mirror length.	0.84
nshells	1	Number of nested shells to expect	4
m	1	Critical angle of mirrors as multiples of Ni.	1
mirr_thick	m	Thickness of mirror shell surfaces - NOT YET IMPLEMENTED	0

Name	Unit	Description	Default
disk		Flag. If nonzero, insert a disk to block the central area within the innermost mirror.	1
radii	m	Optional vector of radii (length should match nshells)	NULL
R0	1	Reflectivity at Low angles for reflectivity curve approximation	0.99
Qc	$Å^{-1}$	Critical scattering vector	0.021
W	$Å^{-1}$	Width of supermirror cut-off	0.003
alpha	Å	Slope of reflectivity for reflectivity curve approximation	6.07
transmit	1	Fraction of statistics to assign to transmitted beam - NOT YET IMPLEMENTED	0
sigma_m	rad	RMS meridional slope (figure) error of the mirror surfaces; for grazing angles below 10*sigma_m the waviness model of Hansen et al., arXiv:1612.03763 is used	1e-6
sigma_s	rad	RMS sagittal slope (figure) error of the mirror surfaces	250e-6

Links

- Component source code found in file `Conics_PP.comp`.

14.23. The `FlatEllipse_finite_mirror` McStas Component

Date: 2022-2023
NMO (nested mirror optic) modules

Identification

- **Author:** Christoph Herb, TUM
- **Origin:** TUM
- **Date:** 2022-2023

Description

Simulates NMO (nested mirror optic) modules as concevied by Böni et al., see Christoph Herb et al., Nucl. Instrum. Meth. A 1040, 1671564 (1-18) 2022.
The component relies on an updated version of conic.h from MIT.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
sourceDist	m	Distance used for calculating the spacing of the mirrors, 0: use LStart	0
LStart	m	Left focal point	0.6
LEnd	m	Right focal point	0.6
lStart	m	z-Value of the mirror start	0.0
lEnd	m	z-Value of the mirror end	0.0
r_0	m	distance to the mirror at lStart	0.02076
nummirror	1	number of mirrors	9
mf		mvalue of the inner side of the coating, m>10 results in perfect reflections	4
mb		mvalue of the outer side of the coating, m>10 results in perfect reflections	0
R0	1	Low-angle reflectivity	0.99
Qc	$Å^{-1}$	critical scattering vector	0.021
W	$Å^{-1}$	width of supermirror cutoff	0.003
alpha	Å	Slope of reflectivity	6.07
mirror_width	m	width of the individual mirrors	0.003
mirror_sidelength	m	side length of the individual mirrors	1
doubleReflections	1	binary value determining whether the mirror backside is reflective	0
rfront_inner_file	str	file of distances to the optical axis of the individual mirrors	"NULL"

Links

- Component source code found in file `FlatEllipse_finite_mirror.comp`.
- Christoph Herb et al., Nucl. Instrum. Meth. A 1040, 1671564 (1-18) 2022.

14.24. The `Mirror_conic` McStas Component

Single flat mirror plate using the conic.h reflection routines (test component)

Identification

- **Author:** Peter Willendrup
- **Origin:** DTU
- **Date:** October 2026

Description

Test component for the reflection routines in conic.h (ConicTracer), which are used by e.g. Conics_*, FlatEllipse_finite_mirror and NMO. It models a single flat mirror so that the results can be compared directly with Mirror.comp.

The mirror lies in the local y-z plane at x=0, extending from z=0 to z=length and from y=-yheight/2 to y=yheight/2. Only neutrons arriving from the x<0 side (vx>0) interact; everything else passes unaffected. Each neutron interacts at most once.

conic=0: the mirror is a conic.h FlatSurf and is handled by reflectNeutronFlat(), which reflects with probability R and otherwise transmits the neutron unaffected. Compare with Mirror(transmit=1). conic=1: the mirror is a conic.h ConicSurf cylinder of (very large) radius 1 km, handled by reflectNeutronConic(), which always reflects and weights by R. Compare with Mirror(transmit=0). A Gaussian figure (slope) error can be applied via sigma_m and sigma_s; the default is 0 here, whereas the Conics_* components default to the conic.h values of 1 and 250 urad.

Example: Mirror_conic(length=1, yheight=0.1, m=2)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
length	m	Length of the mirror along z	1
yheight	m	Height of the mirror along y	0.1
R0	1	Low-angle reflectivity	0.99
Qc	$Å^{-1}$	Critical scattering vector	0.021
alpha	Å	Slope of reflectivity	6.07
m	1	m-value of material	2
W	$Å^{-1}$	Width of supermirror cut-off	0.003
conic	1	0: use FlatSurf/reflectNeutronFlat(), 1: use ConicSurf/reflectNeutronConic()	0
sigma_m	rad	RMS meridional slope (figure) error, conic=1 only; for grazing angles below 10*sigma_m the waviness model of Hansen et al., arXiv:1612.03763 is used	0
sigma_s	rad	RMS sagittal slope (figure) error, conic=1 only	0

Links

- Component source code found in file `Mirror_conic.comp`.

14.25. The `Commodus_I3` McStas Component

ISIS Moderators (Tested with McStas 3.1 (Windows))

Identification

- **Author:** G. Skoro, based on ViewModerator4 from S. Ansell
- **Origin:** ISIS
- **Date:** July 2022

Description

Produces a neutron distribution at the ISIS TS1 or TS2 corresponding moderator front face position. The Face argument determines which TS1 or TS2 beamline is to be sampled by using corresponding file. Neutrons are created having a range of energies determined by the E0 and E1 arguments. Trajectories are produced such that they pass through the moderator face (defined by modXsize and modZsize) and a focusing rectangle (defined by xw, yh and dist). — HOW TO USE —

```
Example:   Commodus_I3(Face="TS1verBase2016_LH8020_newVM-var_South04_Merlin.mcstas",
modXsize = 0.12, modZsize = 0.12, xw = 0.094, yh = 0.094, dist = 1.6)
```

MERLIN simulation; TS1 baseline model. In this example, xw and yh are chosen to be identical to the shutter opening dimension. dist = 1.6 is the real distance to the shutter front face: (This is TimeOffset value (=160 [cm]) from TS1verBase2016_LH8020_newVM-var_South04_Merlin.mcstas file.)

N.B. Absolute normalization: The result of the Mc-Stas simulation will show neutron intensity for beam current of 1 uA.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
Face	string	TS1 (or TS2) instrument McStas filename	"TS1_S04_Merlin.m
E0	meV	Lower edge of energy distribution	
E1	meV	Upper edge of energy distribution	
modPosition	1	Defines the initial surface for neutron distribution production. Possible values = 0 (at moderator) or 1 (at shutter insert)	0
dist	m	Distance from moderator surface to the focusing rectangle	1.7

Name	Unit	Description	Default
verbose	int	Flag to output debugging information	0
beamcurrent	uA	ISIS beam current	1
modXsize	m	Moderator width	0.12
modZsize	m	Moderator height	0.12
xw	m	Width of focusing rectangle	0.094
yh	m	Height of focusing rectangle	0.094

Links

- Component source code found in file `Commodus_I3.comp`.

14.26. Contributed choppers and collimators

14.27. The `FermiChopper_ILL` McStas Component

Fermi Chopper with rotating frame.

Identification

- **Author:** M. Poehlmann, C. Carbogno, H. Schober, E. Farhi
- **Origin:** ILL Grenoble / TU Muenchen
- **Date:** May 2002

Description

Models a Fermi chopper with optional supermirror coated blades supermirror facilities may be disabled by setting m = 0, R0=0 Slit packages are straight. Chopper slits are separated by an infinitely thin absorbing material. The effective transmission (resulting from fraction of the transparent material and its transmission) may be specified. The chopper slit package width may be specified through the total width 'xwidth' of the full package or the width 'w' of each single slit. The other parameter is calculated by: xwidth = nslit*w.

Example: FermiChopper_ILL(phase=-50.0, radius=0.04, nu=100,

```
yheight=0.08, w=0.00022475, nslit=200.0, R0=0.0,
Qc=0.02176, alpha=2.33, m=0.0, length=0.012, eff=0.95,
```

zero_time=0)

```
Markus Poehlmann     <Markus.Poehlmann@ph.tum.de>
Christian Carbogno   <carbogno@ph.tum.de>
and Helmut Schober   <schober@ill.fr>
```

%VALIDATION Apr 2005: extensive external test, most problems solved (cf. 'Bugs') Validated by: K. Lieutenant

limitations: no blade width used

%BUGS - overestimates peak width for long wavelengths - does not give the right pulse position, shape and width for slit widths below 0.1 mm - fails sometimes when using MPI

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
phase	deg	chopper phase at t=0	0
radius	m	chopper cylinder radius	0.04
nu	Hz	chopper frequency	100
yheight	m	Height of chopper	0.08
w	m	width of one chopper slit	0.00022475
nslit	1	number of chopper slits	200.0
R0	1	low-angle reflectivity	0.0
Qc	$Å^{-1}$	critical scattering vector	0.02176
alpha	Å	slope of reflectivity	2.33
m	1	m-value of material. Zero means completely absorbing.	0.0
W	$Å^{-1}$	width of supermirror cut-off	2e-3
length	m	channel length of the Fermi chopper	0.012
eff	1	efficiency = transmission x fraction of transparent material	0.95
zero_time	1	set time to zero: 0: no 1: once per half cycle	0
xwidth	m	optional total width of slit package	0
verbose	1	optional flag to display component statistics, use 1 or 3 (debuging)	0

Links

- Component source code found in file `FermiChopper_ILL.comp`.

14.28. The `Fermi_chop2a` McStas Component

SNS Fermi Chopper as used in SNS_ARCS

Identification

- **Author:** Garrett Granroth

- **Origin:** SNS Oak Ridge,TN
- **Date:** 6 Feb 2005

Description

Models an SNS Fermi Chopper, used in the SNS_ARCS instrument model.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
len	m	slit package length	
w	m	slit package width	
nu	Hz	frequency	
delta	sec	time from edge of chopper to center Phase angle	
tc	sec	time when desired neutron is at the center of the chopper	
ymin	m	Lower y bound	
ymax	m	Upper y bound	
nchan	1	number of channels in chopper	
bw	m	blade width	
blader	m	blade radius	

Links

- Component source code found in file `Fermi_chop2a.comp`.

14.29. The `MultiDiskChopper` McStas Component

Based on DiskChopper (Revision 1.18) by Peter Willendrup (2006), which in turn is based on Chopper (Philipp Bernhardt), Jitter and beamstop from work by Kaspar Hewitt Klenoe (jan 2006), adjustments by Rob Bewey (march 2006)

Identification

- **Author:** Markus Appel
- **Origin:** ILL / FAU Erlangen-Nuernberg
- **Date:** 2015-10-19

Description

Models a disk chopper with a freely configurable slit pattern. For simple applications, use the DiskChopper component and see the component manual example of DiskChopper GROUPing. If the chopper slit pattern should be dynamically configurable or a complicated pattern is to be used as first chopper on a continuous source, use this component.

Width and position of the slits is defined as a list in string parameters so they can easily be taken from instrument parameters. The chopper axis is located on the y axis as defined by the parameter delta_y. When the chopper is the first chopper after a continuous (i.e. time-independent) source, the parameter isfirst should be set to 1 to increase Monte-Carlo efficiency.

Examples (see parameter definitions for details): Two opposite slits with 10 and 20deg opening, with the 20deg slit in the beam at t=0.02: MultiDiskChopper(radius=0.2, slit_center="0;180", slit_width="10;20", delta_y=-0.1, nu=302, nslits=2, phase=180, delay=0.02)

First chopper on a continuous source, creating pulse trains for one additional revolution before and after the revolution at t=0: MultiDiskChopper(radius=0.2, slit_center="0;180", slit_width="10;20", delta_y=-0.1, nu=302, nslits=2, phase=180, isfirst=1, nrev=1)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
slit_center	string	(deg) Angular position of the slits (similar to slit_width)	"0 180"
slit_width	string	(deg) Angular width of the slits, given as list in a string separated by space ' ', comma ',', underscore '_' or semicolon ';'. Example: "0;20;90;135;270"	"10 20"
nslits		Number of slits to read from slit_width and slit_center	2
delta_y	m	y-position of the chopper rotation axis. If the chopper is located above the guide (delta_y>0), the coordinate system will be mirrored such that the created pulse pattern in time is the same as for delta_y<0. A warning will be printed in this case.	-0.3
nu	Hz	Rotation speed of the disk, the sign determines the direction.	0

Name	Unit	Description	Default
nrev		When isfirst=1: Number of *additional* disk revolutions before *and* after the one around t=delay to distribute events on. If set to 2 for example, there will be 2 leading, 1 central, and 2 trailing revolutions of the disk (2*nrev+1 in total).	0
ratio		When isfirst=1: Spacing of the additional revolutions from the parameter nrev from the central revolution.	1
jitter	s	Jitter in the time phase.	0
delay	s	Time delay of the chopper clock.	0
isfirst	0/1	Set to 1 for the first chopper after a continuous source. The neutron events will be shifted in time to pass the component (with adapted weight).	0
phase	deg	Phase angle located on top of the disk at t=delay (see below).	0
radius	m	Outer radius of the disk	0.375
equal	0/1	When isfirst=1: If 0, the neutron events will be distributed between different slits proportional to the slit size. If 1, the events will be distributed such that each slit transmits the same number of events. This parameter can be used to achieve comparable simulation statistics over different pulses when simulating small and large slits together.	0
abs_out		If 1, absorb all neutrons outside the disk diameter.	0
verbose	0/1	Set to 1 to display more information during the simulation.	0

Links

- Component source code found in file `MultiDiskChopper.comp`.

14.30. The `StatisticalChopper` McStas Component

Statistical (correlation) Chopper

Identification

- **Author:** C. Monzat/E. Farhi/S. Rozenkranz
- **Origin:** ILL
- **Date:** Nov 10th, 2009

Description

This component is a statistical chopper based on a pseudo random sequence that the user can specify. Inspired from DiskChopper, it should be used with SatisticalChopper_Monitor component.

Example: StatisticalChopper(nu=1487*60/255) use default sequence

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	m	Radius of the disc	0.5
yheight	m	Slit height (if = 0, equal to radius). Auto centering of beam at half height.	0
nu	Hz	Frequency of the Chopper, omega=2*PI*nu (algebraic sign defines the direction of rotation)	
jitter	s	Jitter in the time phase	0
delay	s	Time 'delay'.	0
isfirst	0/1	Set it to 1 for the first chopper position in a cw source (it then spreads the neutron time distribution)	0
abs_out	0/1	Absorb neutrons hitting outside of chopper radius?	1
phase	deg	Angular 'delay' (overrides delay)	0
verbose	1	Set to 1 to display Disk chopper configuration	0
n_pulse	1	Number of pulses (Only if isfirst)	1
sequence	str	Slit sequence around disk, with 0=closed, 1=open positions. NULL or "" will set a default sequence.	"NULL"

Links

- Component source code found in file `StatisticalChopper.comp`.

- R. Von Jan and R. Scherm. The statistical chopper for neutron time-of-flight spectroscopy. Nuclear Instruments and Methods, 80 (1970) 69-76.

14.31. The `StatisticalChopper_Monitor` McStas Component

Monitor designed to compute the autocorrelation signal for the Statistical Chopper

Identification

- **Author:** C. Monzat/E. Farhi
- **Origin:** ILL
- **Date:** 2009

Description

This component is a time sensitive monitor which calculates the cross correlation between the pseudo random sequence of a statistical chopper and the signal received. It mainly uses fonctions of component Monitor_nD. It is possible to use the various options of the Monitor_nD but the user MUST NOT specify "time" in the options. Auto detection of the time limits is possible if the user chooses tmin=>tmax.

StatisticalChopper_Monitor(options ="banana bins=500, abs theta limits=[5,105],bins=1000")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
comp	StatisticalChopper	required name of the component monitored	
tmin	s	minimal time of detection	0.0
tmax	s	maximal time of detection	0.0

Links

- Component source code found in file `StatisticalChopper_Monitor.comp`.
- R. Von Jan and R. Scherm. The statistical chopper for neutron time-of-flight spectroscopy. Nuclear Instruments and Methods, 80 (1970) 69-76.

14.32. The `Vertical_T0a` McStas Component

Vertical T0-chopper as used in SNS_ARCS

Identification

- **Author:** Garrett Granroth
- **Origin:** SNS Oak Ridge,TN
- **Date:** 2 NOV 2004

Description

Vertical T0-chopper as used in SNS_ARCS

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
len	m	length of slot	
w1	m	center width	
w2	m	edgewidth	
nu	Hz	frequency	
delta	s	time from edge of chopper to center Phase angle	
tc	s	time when desired neutron is at the center of the chopper	
ymin	m	Lower y bound	
ymax	m	Upper y bound	

Links

- Component source code found in file `Vertical_T0a.comp`.

14.33. The `Collimator_ROC` McStas Component

Radial Oscillationg Collimator (ROC)

Identification

- **Author:** Thomas C Hansen
- **Origin:** ILL (Dif/<a href="http://www.ill.fr/YellowBook/
- **Date:** 15 May 2000

Description

ESCRIPTION

This is an implementation of an Ideal radial oscillating collimator, which is usually placed between a polycrystalline sample and a linear curved position sensitive detector on a 2-axis diffractometer like D20. The transfer function has been implemented analytically, as this is much more efficient than doing Monte-Caqrlo (MC) choices and to absorb many neutrons on the absorbing blades. The function is basically triangular (except for rather 'exotic' focuss apertures) and depends on the distance from the projection of the focus centre to the plane perpendicular to the collimator planes to the intersection of the same projection of the velocity vector of the neutron with a line that is perpendicular to it and containing the same projection of the focus centre. The oscillation is assumed to be absolutely regular and so shading each angle the same way. All neutrons not hitting the collimator core will be absorbed.

Example: Collimator_ROC(ROC_pitch=5, ROC_ri=0.15, ROC_ro=0.3, ROC_h=0.15, ROC_ttmin=-25, ROC_ttmax=135, ROC_sign=-1)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
ROC_pitch	deg	Angular pitch between the absorbing blades	1
ROC_ri	m	Inner radius of the collimator	0.4
ROC_ro	m	Outer radius of the collimator	1.2
ROC_h	m	Height of the collimator	0.153
ROC_ttmin	deg	Lower scattering angle limit	0
ROC_ttmax	deg	Higher scattering angle limit	100
ROC_sign	1	Chirality/takeoff sign	1

Links

- Component source code found in file `Collimator_ROC.comp`.
- D20 diffractometer at the ILL

14.34. The `Exact_radial_coll` McStas Component

An exact radial Soller collimator.

Identification

- **Author:** Roland Schedler <roland.schedler at hmi.de>

- **Origin:** HMI
- **Date:** October 2006

Description

Radial Soller collimator with rectangular opening, specified length and specified foil thickness. The collimator is made of many trapezium shaped nslit stacked radially. The nslit are separated by absorbing foils, the whole stuff is inside an absorbing housing. The component should be positioned at the radius center. The model is exact. The neutron beam outside the collimator area is transmitted unaffected.

Example: Exact_radial_coll(theta_min=-5, theta_max=5, nslit=100,

```
radius=1.0, length=.3, h_in=.2, h_out=.3, d=0.0001)
```

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
theta_min	deg	Minimum Theta angle for the radial setting	-5
theta_max	deg	Maximum Theta angle for the radial setting	5
nslit	1	Number of channels in the theta range	100
radius	m	Inner radius (focus point to foil start point).	1.0
length	m	Length of the foils / collimator	.5
h_in	m	Input window height	.3
h_out	m	Output window height	.4
d	m	Thickness of the absorbing foils	0.0001
verbose	0/1	Gives additional information	0

Links

- Component source code found in file `Exact_radial_coll.comp`.

14.35. The `CavitiesIn` McStas Component

Slit - sorting in channels

Identification

- **Author:** Henrich Frielinghaus
- **Origin:** JCNS - FZ-Juelich
- **Date:** Oct 2007

Description

This routine sorts the 'full' neutron beam (given by xw,yw) in xc,yc channels. These can be imagined as cavities. CavitiesOut sorts these channels back to normal coordinates.
Example: Slit(xw=0.05, yw=0.05, xc=4, yc=1)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xw	m	width in X-dir	0.05
yw	m	width in Y-dir	0.05
xc	m	channels in X-dir	1
yc	m	channels in Y-dir	1

Links

- Component source code found in file `CavitiesIn.comp`.

14.36. The `CavitiesOut` McStas Component

Slit - sorting in channels

Identification

- **Author:** Henrich Frielinghaus
- **Origin:** JCNS - FZ-Juelich
- **Date:** Oct 2007

Description

This routine sorts the 'full' neutron beam (given by xw,yw) in xc,yc channels. These can be imagined as cavities. CavitiesOut sorts these channels back to normal coordinates.
Example: Slit(xw=0.05, yw=0.05, xc=4, yc=1)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xw	m	width in X-dir (full width, might be larger for trumpets)	0.05
yw	m	width in Y-dir (full width, might be larger for trumpets)	0.05
xc	1	Number of x-channels	1
yc	1	Number of y-channels	1

Links

- Component source code found in file `CavitiesOut.comp`.

14.37. The `multi_pipe` McStas Component

multi-pipe circular slit.

Identification

- **Author:** Uwe Filges
- **Origin:** PSI
- **Date:** March, 2005

Description

No transmission around the slit is allowed.
example for an input file # user defined geometry

```
# x(i) y(i) r(i) [m]
0.02 0.03 0.01
-0.04 0.015 0.005
```

warning: at least two values must be in the file
Example: multi_pipe(xmin=-0.01, xmax=0.01, ymin=-0.01, ymax=0.01)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename	str	define user table of holes	0
xmin	m	Lower x bound	
xmax	m	Upper x bound	
ymin	m	Lower y bound	
ymax	m	Upper y bound	
radius	m	radius of a single hole	0.0
gap	m	distance between holes	0.0
thickness	m	thickness of the pipe	0.0
xwidth	m	Width of slit plate. Overrides xmin,xmax.	0
yheight	m	Height of slit plate. Overrides ymin,ymax.	0

Links

- Component source code found in file `multi_pipe.comp`.

14.38. Contributed filters and windows

14.39. The `Al_window` McStas Component

Aluminium window in the beam

Identification

- **Author:** S. Roth
- **Origin:** FRM-II
- **Date:** Jul 31, 2001

Description

Aluminium window in the beam
Example: Al_window(thickness=0.002)
%BUGS Only handles the absorption, and window has infinite width/height

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
thickness	m	Al Window thickness	0.001

Links

- Component source code found in file `Al_window.comp`.

14.40. The `Filter_graphite` McStas Component

Pyrolytic graphite filter (analytical model)

Identification

- **Author:** Thomas C Hansen
- **Origin:** ILL
- **Date:** 07 March 2000

Description

ESCRIPTION

This rectangular pyrolytic graphite filter uses an analytical model with linear interpolation on the neutron wavelength This type of filter is e.g. used to supress higher harmonics, such as the 1.2 Å contribution to the 2.4 Å obtained by a highly oriented pyrolytic graphite (HOPG) monochromator.

Example: Filter_graphite(xmin=-.05, xmax=.05, ymin=-.05, ymax=.05, length=1.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xmin	m	Lower x bound	-0.16
xmax	m	Upper x bound	0.16
ymin	m	Lower y bound	-0.16
ymax	m	Upper y bound	0.16
length	m	Thickness of graphite plate	0.05
xwidth	m	Width of filter. Overrides xmin,xmax.	0
yheight	m	Height of filter. Overrides ymin,ymax.	0

Links

- Component source code found in file `Filter_graphite.comp`.

14.41. The `Sapphire_Filter` McStas Component

Sapphire filter at 300K

Identification

- **Author:** Jonas Okkels Birk (based upon Filteg_Graphite by Thomas C Hansen (2000))
- **Origin:** PSI
- **Date:** 6 December 2006

Description

ESCRIPTION

This sapphire filter, defined by two identical rectangular opening apertures, is based upon an absorption function absorp=exp(L*A+C*(exp(-(B/L^2+D/L^4)))) decribed by Freund (1983) Nucl. Instrum. Methods, A278, 379-401. The defaultvalues is for Sapphire at 300K as found by measurement by (Mildner & Lamaze (1998) J. Appl. Cryst. 31,835-840 (http://journals.iucr.org/j/issues/1998/06/00/hw0070/hw0070bdy.html#BB4)). It is possble to ajust the formular to other materials or temperatures by typing in other parameters. The transmission in sapphire is only lightly demependt on temperature, but The formular is only cheked at wavelenths between 0.5 and 14 Å (0.4 meV 0.3 eV). The filter is for example used in the Eiger beamline at PSI to cut of high energies.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xmin	m	Lower x bound	-0.16
xmax	m	Upper x bound	0.16
ymin	m	Lower y bound	-0.16
ymax	m	Upper y bound	0.16
len	m	Thickness of filter	0.1
A	m^{-1} $Å^{-1}$	Coefficient of the attenuation fit mu(lambda) = A*lambda + C*(1-exp(-B/lambda^2 - D/lambda^4))	0.8116
B	$Å^2$	Coefficient of the attenuation fit mu(lambda) = A*lambda + C*(1-exp(-B/lambda^2 - D/lambda^4))	0.1618
C	m^{-1}	Coefficient of the attenuation fit mu(lambda) = A*lambda + C*(1-exp(-B/lambda^2 - D/lambda^4))	21.24

Name	Unit	Description	Default
D	Å^4	Coefficient of the attenuation fit mu(lambda) = A*lambda + C*(1-exp(-B/lambda^2 - D/lambda^4))	0.1291

Links

- Component source code found in file `Sapphire_Filter.comp`.

14.42. Contributed lenses and mirrors

14.43. The `Lens` McStas Component

Refractive lens with absorption, incoherent scattering and surface imperfection.

Identification

- **Author:** C. Monzat/E. Farhi/S. Desert/G. Euzen
- **Origin:** ILL/LLB
- **Date:** 2009

Description

Refractive Lens with absorption, incoherent scattering and surface imperfection. Geometry may be: spherical (use r1 and r2 to specify radius of curvature),

```
planar    (use phiy1 and phiy2 to specify rotation angle of plane w.r.t beam)
```

parabolic (use focus1 and focus2 as optical focal length).

Optionally, you can specify the 'geometry' parameter as a OFF/PLY file name. The complex geometry option handles any closed non-convex polyhedra. It computes the intersection points of the neutron ray with the object transparently, so that it can be used like a regular sample object. It supports the OFF (incl. NOFF, COFF), PLY and STL file formats. Such files may be generated from XYZ data using: qhull < coordinates.xyz Qx Qv Tv o > geomview.off or powercrust coordinates.xyz and viewed with geomview or java -jar jroff.jar (see below).

The lens cross-section is seen as a 2*radius disk from the beam Z axis, except when a 'geometry' file is given. Usually, you should stack more than one of these to get a significant effect on the neutron beam, so-called 'compound refractive lens'.

The focal length for N lenses with focal 'f' is f/N, where f=R/(1-n)

```
and R = r/2   for a spherical lens with curvature radius 'r'
```

R = focus for a parabolic lens with focal of the parabola and n = sqrt(1-(lambda*lambda*rho*bc/PI)) with bc = sqrt(fabs(sigma_coh)*100/4/PI)*1e-5 rho = density*6.02214179*1e23*1e-24/weight

Common materials: Should have high coherent, and low incoherent and absorption cross sections

```
Be:            density=1.85,  weight=9.0121, sigma_coh=7.63,  sigma_inc=0.0018,sigma_abs=0.0076
Pb:            density=11.115,weight=207.2,  sigma_coh=11.115,sigma_inc=0.003, sigma_abs=0.171
Pb206:                                       sigma_coh=10.68, sigma_inc=0    , sigma_abs=0.03
Pb208:                                       sigma_coh=11.34, sigma_inc=0    , sigma_abs=0.00048
Zr:            density=6.52,  weight=91.224, sigma_coh=6.44,  sigma_inc=0.02,  sigma_abs=0.185
Zr90:                                        sigma_coh=5.1,   sigma_inc=0    , sigma_abs=0.011
Zr94:                                        sigma_coh=8.4,   sigma_inc=0    , sigma_abs=0.05
Bi:            density=9.78,  weight=208.98, sigma_coh=9.148, sigma_inc=0.0084,sigma_abs=0.0338
Mg:            density=1.738, weight=24.3,   sigma_coh=3.631, sigma_inc=0.08,  sigma_abs=0.063
MgF2:          density=3.148, weight=62.3018,sigma_coh=11.74, sigma_inc=0.0816,sigma_abs=0.0822
diamond:       density=3.52,  weight=12.01,  sigma_coh=5.551, sigma_inc=0.001, sigma_abs=0.0035
```

Quartz/silica: density=2.53, weight=60.08, sigma_coh=10.625,sigma_inc=0.0056,sigma_abs=0.1714

```
Si:            density=2.329, weight=28.0855,sigma_coh=2.1633,sigma_inc=0.004, sigma_abs=0.171
Al:            density=2.7,   weight=26.98,  sigma_coh=1.495, sigma_inc=0.0082,sigma_abs=0.231
```

perfluoropolymer(PTFE/Teflon/CF2):

```
density=2.2,   weight=50.007, sigma_coh=13.584,sigma_inc=0.0026,sigma_abs=0.0227
```

Organic molecules with C,O,H,F

Example: Lens(r1=0.025,r2=0.025, thickness=0.001,radius=0.0150)

%BUGS parabolic shape is not fully validated yet, but should do still...

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
r1	m	radius of the first circle describing the lens. r>0 means concave face, r<0 means convex, r=0 means plane	0
r2	m	radius of the second circle describing the lens r>0 means concave face, r<0 means convex, r=0 means plane	0
focus1	m	focal of the first parabola describing the lens	0

Name	Unit	Description	Default
focus2	m	focal of the second parabola describing the lens	0
phiy1	deg	angle of plane1 (r1=0) around y vertical axis	0
phiy2	deg	angle of plane2 (r2=0) around y vertical axis	0
thickness	m	thickness of the lens between its two surfaces	0.001
radius	m	radius of the lens section, e.g. beam size.	0.015
sigma_coh	barn	coherent cross section	11.74
sigma_inc	barn	incoherent cross section	0.0816
sigma_abs	barn	thermal absorption cross section	0.0822
density	g/cm^3	density of the material for the lens	3.148
weight	g/mol	molar mass of the material	62.3018
p_interact	1	MC Probability for scattering the ray; otherwise transmit	0.1
focus_aw	deg	vertical angle to forward focus after scattering	10
focus_ah	deg	horizontal angle to forward focus after scattering	10
RMS	Å	root mean square roughness of the surface	0
geometry	str	Name of an OFF, PLY or STL file for complex geometry. An OFF file may be generated from XYZ coordinates using qhull/powercrust	0

Links

- Component source code found in file `Lens.comp`.
- M. L. Goldberger et al, Phys. Rev. 71, 294 - 310 (1947)
- Sears V.F. Neutron optics. An introduction to the theory of neutron optical phenomena and their applications. Oxford University Press, 1989.
- H. Park et al. Measured operationnal neutron energies of compound refractive lenses. Nuclear Instruments and Methods B, 251:507-511, 2006.
- J. Feinstein and R. H. Pantell. Characteristics of the thick, compound refractive lens. Applied Optics, 42 No. 4:719-723, 2001.
- Geomview and Object File Format (OFF)

- Java version of Geomview (display only) jroff.jar
- qhull
- powercrust

14.44. The `Lens_simple` McStas Component

Rectangular/circular slit with parabolic/spherical LENS.

Identification

- **Author:** Henrich Frielinghaus
- **Origin:** FZ Juelich
- **Date:** June 2010

Description

A simple rectangular or circular slit. You may either specify the radius (circular shape), or the rectangular bounds. No transmission around the slit is allowed.

At the slit position there is/are also a LENS or multiple LENSES present. Spherical/parabolic abberation is taken into account in a simplified manner. The focal point becomes a function of the distance ss from the origin of the lens where the ray hits the lens (plane). With this focal distance the refraction is calculated based on simple geometrical optics (as tought already in school).

The approximation gets wrong when the distance ss is too big, and total reflection becomes possible. Then the corrections of the focal distance are strong (see foc*= ... lines. When this correction factor is too close to zero, the corrections become highly wrong).

The advantage of this routine is the simplicity which makes the simulation very fast - especially for multiple lenses. The argument for this way of treating the lenses is that the collimation and detector distance are large (say 20m) compared to the lens thickness (say 2-5cm) and the lens array thickness (say max. 1m).

For lens arrays one still can simulate several of these simplified lenses instead of an exact treatment (because 5cm<<1m). The author believes that this medium detailed treatment meets usually the desired accuracy.

Example: Lens_simple(rho=5.16e10,Rc=0.0254,Nl=7,xmin=-0.01,xmax=0.01,ymin=-0.01,ymax=0.01) Lens_simple(rho=5.16e10,Rc=0.0254,Nl=7,radius=0.01) Lens_simple(rho=5.16e10,Rc=0.0254,d0=0.002) ^^^^^^^^^^^ last example includes absorption of MgF2-lens.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
rho	m^{-2}	Scattering length density	5.16e14
Rc	m	Radius (concave: Rc>0) of biconcave lens	0.02
Nl	1	Number of single lenses	7.0
parab		Switch (not 0 -> parabolic, for 0 spherical)	1.1
xmin	m	Lower x bound	0
xmax	m	Upper x bound	0
ymin	m	Lower y bound	0
ymax	m	Upper y bound	0
radius	m	Radius of slit in the z=0 plane, centered at Origin	0
SigmaAL	1/(m*Å)	Absorption cross section per lambda (wavelength)	0.141
d0	m	Minimum thickness in the centre of the lens	0.002

Links

- Component source code found in file `Lens_simple.comp`.

14.45. The `Mirror_Curved_Bispectral` McStas Component

Single mirror plate that is curved and fits into an elliptic guide.

Identification

- **Author:** Henrik Jacobsen
- **Origin:** RNBI
- **Date:** April 2012

Description

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
reflect	q($Å^{-1}$ R(0-1)	(str) Name of relfectivity file. Format	0
focus_s	m	first focal point of ellipse in ABSOLUTE COORDINATES	
focus_e	m	second focal point of ellipse in ABSOLUTE COORDINATES	
mirror_start	m	start of mirror in ABSOLUTE COORDINATES	
guide_start	m	start of guide in ABSOLUTE COORDINATES	
yheight	m	the height of the mirror when not in the guide	
smallaxis	m	the smallaxis of the guide	
length	m	the length of the mirror R0=0.99: [] low angle reflectivity	
m		m-value of material	
transmit	0/1	0: non reflected neutrons are absorbed. 1: non reflected neutrons pass through	0
substrate_thickness	m	thickness of substrate (for absorption)	0.0005
coating_thickness	m	thickness of coating (for absorption)	10e-6
theta_1	deg	angle of mirror wrt propagation direction at start of mirror	1.25
theta_2	deg	angle of mirror wrt propagation direction at center of mirror	1.25
theta_3	deg	angle of mirror wrt propagation direction at end of mirror	1.25

Links

- Component source code found in file `Mirror_Curved_Bispectral.comp`.

14.46. The `Mirror_Elliptic` McStas Component

Elliptical mirror.

Identification

- **Author:** Sylvain Desert
- **Origin:** LLB
- **Date:** 2007

Description

Models an elliptical mirror. The reflectivity profile is given by a 2-column reflectivity free text file with format [q(Å-1) R(0-1)]. The component is centered on the ellipse.

Example: Mirror_Elliptic(reflect="supermirror_m3.rfl", focus=6.6e-4, interfocus = 8.2, yheight = 0.0002, zmin=-3.24, zmax=-1.49)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
reflect	q($Å^{-1}$ R(0-1)	(str) Reflectivity file name. Format	0
focus	m	focal length (m)	6.6e-4
interfocus	m	Distance between the two elliptical focal points	8.2
yheight	m	height of the mirror	2e-4
zmin	m	z-axis coordinate of ellipse start	0
zmax	m	z-axis coordinate of ellipse end	0
R0	1	Low-angle reflectivity	0.99
Qc	$Å^{-1}$	Critical scattering vector	0.0219
alpha	Å	Slope of reflectivity	6.07
m	1	m-value of material. Zero means completely absorbing.	1.0
W	$Å^{-1}$	Width of supermirror cut-off	0.003

Links

- Component source code found in file `Mirror_Elliptic.comp`.

14.47. The `Mirror_Elliptic_Bispectral` McStas Component

Single mirror plate that is curved and fits into an elliptic guide.

Identification

- **Author:** Henrik Jacobsen
- **Origin:** RNBI
- **Date:** April 2012

Description

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
reflect	q($Å^{-1}$ R(0-1)	(str) Name of relfectivity file. Format	0
focus_start_w	m	Horizontal position of first focal point of ellipse in ABSOLUTE COORDINATES	
focus_end_w	m	Horizontal position of second focal point of ellipse in ABSOLUTE COORDINATES	
focus_start_h	m	Vertical position of first focal point of ellipse in ABSOLUTE COORDINATES	
focus_end_h	m	VErtical position of second focal point of ellipse in ABSOLUTE COORDINATES	
mirror_start	m	start of mirror in ABSOLUTE COORDINATES	
m		m-value of material	
smallaxis_w	m	Small-axis dimension of horizontal ellipse	
smallaxis_h	m	Small-axis dimension of vertical ellipse	
length	m	the length of the mirror R0=0.99: [] low angle reflectivity	
transmit	0/1	0: non reflected neutrons are absorbed. 1: non reflected neutrons pass through	0
substrate_thickness	m	thickness of substrate (for absorption)	0.0005
coating_thickness	m	thickness of coating (for absorption)	10e-6

Links

- Component source code found in file `Mirror_Elliptic_Bispectral.comp`.

14.48. The `Mirror_Parabolic` McStas Component

Parabolic mirror.

Identification

- **Author:** Sylvain Desert
- **Origin:** LLB
- **Date:** 2007

Description

Models a parabolic mirror. The reflectivity profile is given by a 2-column reflectivity free text file with format [q(Å-1) R(0-1)].

Example: Mirror_Parabolic(reflect="supermirror_m3.rfl", xwidth = 0.05, xshift=0.05, yheight = 2e-4, focus = 6.6e-4)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
reflect	q($Å^{-1}$ R(0-1)	(str) Reflectivity file name. Format	0
xwidth	m	width of the illuminated parabola	0.05
xshift	m	distance between the beam centre and the symetric axis of the parabolla	0.019
yheight	m	height of the mirror	2e-4
focus	m	focal length	6.6e-4
R0	1	Low-angle reflectivity	0.99
Qc	$Å^{-1}$	Critical scattering vector	0.0219
alpha	Å	Slope of reflectivity	6.07
m	1	m-value of material. Zero means completely absorbing.	1.0
W	$Å^{-1}$	Width of supermirror cut-off	0.003

Links

- Component source code found in file `Mirror_Parabolic.comp`.

14.49. The `SupermirrorFlat` McStas Component

Model of flat supermirror with parallel mirror surfaces

Identification

- **Author:** Wai Tung Lee
- **Origin:** ESS
- **Date:** September 2024

Description

Calculate the neutron reflection and transmission of a flat supermirror with parallel mirror surfaces. The following can be specified: 1. reflection from either mirror coating or substrate surface, mirror coating can be single-side, double-side, 2. absorber coating: beneath mirror coating, single-side coated, double-side coated, or uncoated, 3. refraction and total reflection at substrate surface, 4. attenuation and internal reflection inside substrate.

note: supermirror name is case-sensitive text entries of parameters below are not sensitive to case default regional spelling is UK, but some paramters accept other reginal spelling

INITIALISATION parameters:

STEP 1: Specify supermirror shape, supermirror coating, absorber and substrate material In this step, supermirror is standing vertically, mirror coating = yz plane with long side along +z.

```
SUPERMIRROR SHAPE -------------------------------------------------
```

1. +x: "top" mirror surface normal, horizontal transverse to beam; +y: vertical up, parallel to mirror surface; +z: longitudinal to beam, parallel to mirror surface; 2. top and bottom mirrors' surface normals are +x and -x, respectively; 3. entrance-side edge normal is -z, exit-side edge normal is +z (beam direction); 3. normals of and points on the remaining edge surfaces at +y and -y are user-specified,

Mirror shape is specified by length, thickness, and the normals of and points on the edge surfaces at +y and -y sides.

The two side-edges at +y and -y are taken to be symmetric about the xz-plane, only one needs to be specified. Use InitialiseStdSupermirrorFlat_detail if not symmetric.

```
SUPERMIRROR COATING -----------------------------------------------
```

Mirror reflectivity parameters are specified by name to look up 6 parameters {R0, Qc, alpha, m, W, beta}.

If name = SubstrateSurface, reflectivity is calculated by

```
R = R0 (when q<=Qc), R0*( (q - (q^2 - Qc^2)^1/2) / (q + (q^2 - Qc^2)^1/2) )^2 (when q>Qc), with
```

with Qc = Qc_sub defined in SubstrateSurfaceParameters and SLD defined in SubstrateParameters, otherwise reflectivity is calculated by R = R0 (when q<=Qc), R = R0*0.5*(1-tanh((q - m*Qc)/W))*(1-alpha*(q-Qc)+beta*(q-Qc)*(q-Qc)) (when q>Qc).

specified mirror material name matching one of those defined in "Supermirror_reflective_coating_materials.txt" If the mirror is non-polarising, specify the same name and parameters for spin plus and spin minus.

```
ABSORBER LAYER ----------------------------------------------------
```

Absorber parameters are either specified by name or by parameters L_abs, L_inc and the coating thickness.

```
SUBSTRATE ------------------------------------------------
```

Substrate parameters are specified by name to look up 3 parameters {L_abs, L_inc, SLD}.

specify substrate material name matching one of those defined in "Supermirror_substrate_materials

STEP 2: Orient and position the as-defined supermirror in the McStas module XYZ coordinates.

1. Specify initially a location on the front side of the mirror (see parameter "initial_placement_at_origin" below). The supermirror is shifted so that the selected point coincides with the origin of the McStas module XYZ coordinates 2. Then the orientation and position of the supermirror are specified by the movements in sequence as 1st, tilting about an axis in +y direction at a selected location (see parameters "tilt_y_axis_location" and "tilt_about_y_first_in_degree" below), 2nd, translation, 3rd, rotation about the z-axis of the McStas module XYZ coordinates.

OUTPUT: Supermirror struct with all parameter values entered and initialised User declares "Supermirror supermirror;" or its equivalence, then passes pointer "&supermirror" to function.

End INITIALISATION parameters

RAY-TRACING parameters

FUNCTION IntersectStdSupermirrorFlat INPUT: neutron parameters: w_sm=p, t_sm=t, p_sm=coords_set(x,y,z), v_sm=coords_set(vx,vy,vz), s_sm=coords_set(sx,sy,sz)

```
last_exit_time   [s]       time of last exit from a supermirror, use F_INDETERMINED i
last_exit_point   [m,m,m]   position of last exit from a supermirror, use coords_set(
last_exit_plane   [1]       plane of last exit from a supermirror, use I_INDETERMINED
```

sm: [struct] Supermirror structure OUTPUT:

```
num_intersect:          [1]  number of intersects through supermirror
```

First intersect time, point, plane if there is intersect. User declare one or more parameters, e.g. "int num_intersect; double first_intersect_time; Coords first_intersect_point; int first_intersect_plane;", then passes pointers "&num_intersect, &first_intersect_time, &first_intersect_point, &first_intersect_plane" to function. Pass 0 as pointer if not needed. first_intersect_dtime: [s] time difference from neutron to first intersect first_intersect_dpoint [m,m,m] position difference from neutron to point of first intersect

```
first_intersect_time:   [s]  time of first intersect
```

first_intersect_point: [m,m,m] point of first intersect first_intersect_plane: [1] plane of first intersect RETRUN: sm_Intersected, sm_Missed. sm_Error

FUNCTION StdSupermirrorFlat INPUT: sm [struct] Supermirror structure INPUT & OUTPUT: neutron parameters: w_sm=p, t_sm=t, p_sm=coords_set(x,y,z), v_sm=coords_set(vx, s_sm=coords_set(sx,sy,sz)

```
last_exit_time   [s]       time of last exit from a supermirror, use F_INDETERMINED if not dete
last_exit_point   [m,m,m]   position of last exit from a supermirror, use coords_set(F_INDETERM
last_exit_plane   [1]       plane of last exit from a supermirror, use I_INDETERMINEDif not det
```

RETRUN: sm_Exited, sm_Absorbed. sm_Error sm: [struct Supermirror] Supermirror structure

End RAY-TRACING parameters

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
side_edge_normal	1,1,1	Normal vector of one of the two side-edge surface, use Coords structure, doesn't need to be normalised	{0,+1,0}
side_edge_point	m,m,m	A point on one of the two side-edge surface with its normal specified above, use Coords structure	{0,+0.1,0}
length	m	Supermirror length	0.5
thickness_in_mm	mm	Substrate thickness in mm	0.3
mirror_coated_side	string	Sequential keywords combinations of position and surface property. position: "Both", "Top", "Bottom"; surface property: "Coated", "SubstrateSurface", "NoReflection"; Note: "NotCoated"="Empty"="SubstrateSurface", e.g. "BothCoated", "BottomCoatedTopSubstrate" (case-insensitive.)	"BothCoated"
mirror_spin_plus_material_name	string	mirror spin+ material name in "Supermirror_reflective_coating_materials.txt". (case-insensitive)	"FeSiPlus"
mirror_spin_plus_m	1	mirror spin+ m-value, -1=use default value	3.5
mirror_spin_minus_material_name	string	mirror spin- material name in "Supermirror_reflective_coating_materials.txt". (case-insensitive)	"FeSiMinus"
mirror_spin_minus_m	1	mirror spin- m-value, -1=use default value (useful for spin-)	-1
substrate_material_name	string	substrate material name in "Supermirror_substrate_materials.txt". (Material name is case-insensitive)	"Glass"

Name	Unit	Description	Default
absorber_coated_sides	string	"BothNotCoated", "BothCoated", "TopCoated", "BottomCoated".	"BothNotCoated"
absorber_material_name	string	absorber material name in "Supermirror_absorber_coating_materials.txt" or "Empty", 0="Empty",	"Gd"
absorber_thickness_micrometer	micrometer	absorber coating thickness in micrometer	100
initial_placement_at_origin	string	"TopFrontEdgeCentre","FrontSubstrateCentre","BottomFrontEdgeCentre" (insensitive to case and regional English spelling)	"FrontSubstrateCentre"
tilt_y_axis_location	string	"TopFrontEdge","TopMirrorCentre","TopBackEdge", "FrontSubstrateCentre","SubstrateCentre","BackSubstrateCentre", "BottomFrontEdge","BottomMirrorCentre","BottomBackEdge" (insensitive to case and regional English spelling)	"FrontSubstrateCentre"
tilt_about_y_first_in_deg	degree	First: tilt about the x-axis at the selected location	1.5
translation_second_x	m	Second: translate along x	0
translation_second_y	m	Second: translate along y	0
translation_second_z	m	Second: translate along z	0
rot_about_z_third_in_deg	degree	Third: rotate about z-axis (beam axis) of the McStas module XYZ coordinates	0
tracking	str	"DetailTracking" records the detailed neutron path inside the mirror; any other value disables this	"DetailTracking"

Links

- Component source code found in file `SupermirrorFlat.comp`.

14.50. Contributed monochromators and analysers

14.51. The `Monochromator_2foc` McStas Component

Double bent monochromator with multiple slabs

Identification

- **Author:** Peter Link.
- **Origin:** Uni. Gottingen (Germany)

- **Date:** Feb. 12,1999

Description

Double bent monochromator which uses a small-mosaicity approximation as well as the approximation vy^2 << vz^2 + vx^2. Second order scattering is neglected. For an unrotated monochromator component, the crystal plane lies in the y-z plane (ie. parallel to the beam). When curvatures are set to 0, the monochromator is flat. The curvatures approximation for parallel beam focusing to distance L, with monochromator rotation angle A1 are: RV = 2*L*sin(DEG2RAD*A1); RH = 2*L/sin(DEG2RAD*A1);

Example: Monochromator_2foc(zwidth=0.02, yheight=0.02, gap=0.0005, NH=11, NV=11,

```
mosaich=30, mosaicv=30, r0=0.7, Q=1.8734)
```

Example values for lattice parameters

```
PG       002 DM=3.355 AA (Highly Oriented Pyrolytic Graphite)
PG       004 DM=1.607 AA
```

Heusler 111 DM=3.362 Å (Cu2MnAl)

```
CoFe         DM=1.771 AA (Co0.92Fe0.08)b
Ge       311 DM=1.714 AA
Si       111 DM=3.135 AA
Cu       111 DM=2.087 AA
Cu       002 DM=1.807 AA
Cu       220 DM=1.278 AA
Cu       111 DM=2.095 AA
```

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
reflect	k, R	reflectivity file name of text file as 2 columns	0
zwidth	m	horizontal width of an individual slab	0.01
yheight	m	vertical height of an individual slab	0.01
gap	m	typical gap between adjacent slabs	0.0005
NH	columns	number of slabs horizontal	11
NV	rows	number of slabs vertical	11
mosaich	arcmin	Horisontal mosaic FWHM	30.0
mosaicv	arcmin	Vertical mosaic FWHM	30.0

Name	Unit	Description	Default
r0	1	Maximum reflectivity	0.7
Q	Å^{-1}	Scattering vector	1.8734
RV	m	radius of vertical focussing, flat for 0	0
RH	m	radius of horizontal focussing, flat for 0	0
DM	Å	monochromator d-spacing instead of Q=2*pi/DM	0
mosaic	arcmin	sets mosaich=mosaicv	0
width	m	total width of monochromator	0
height	m	total height of monochromator	0
verbose	0/1	verbosity level	0

Links

- Component source code found in file `Monochromator_2foc.comp`.
- Additional note from Peter Link.

14.52. The `Monochromator_bent` McStas Component

A bent crystal monochromator. Based on the model implemented by Jan Šaroun in NIMA 529 (2004) pp 162-165. An article describing the component has been written and can be seen in Quantum Beam Sci. 2026, 10, 6. https://doi.org/10.3390/qubs10010006 Mosacity and bending radius can be set.

Identification

- **Author:** Daniel Lomholt Christensen <dlc@math.ku.dk> with help from Jan Šaroun
- **Origin:** ILL/NBI
- **Date:** 24 August 2023

Description

This monochromator is an array of crystals, that can be bent. The crystals are placed by the user in the x,y,z pos and rot parameters. The crystal is bent, so that it follows a curve on a cylinder of radius_x. The monochromator lies along the z plane, so when a diffraction angle of theta is desired, it should just be inserted in the ROTATED parameter around the y-axis. Instruments that showcase the use of this component is the "Test_monochromator_bent.instr", and the "ILL_SALSA.instr" under the examples folder. SALSA showcases its complex use in a real instrument, while Test_monochromator_bent makes a simple show of its capabilities.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
zwidth	m	Width of each crystal without bending.	0.2
yheight	m	Height of each crystal without bending.	0.1
xthickness	m	Thickness of each crystal without bending.	0.0005
radius_x	m	Radius of the circle the monochromator bends on in the plane. Can be negative.	2
radius_y	m	Radius of the (very large) circle the monochromator bends on as a side effect of the horizontal bending. The code assumes that it is so small that it does not affect the points of intersection appreciatively of the crystal.	0
plane_of_reflection	"Si^{400}"	The plane of reflection from the material. The list of possible reflections can be seen in the source code.	"Si400"
angle_to_cut_horizontal	degrees	Angle between cut and normal of crystal slab, horizontally	0
mosaicity	arcmin	Gaussian mosaicity of the crystal. Always the horizontal mosaicity	30
mosaic_anisotropy	1	Anisotropy of the mosaicity, changes vertical mosaicity to be mosaic_anisotropy*mosaicity	1
n_crystals	#	Number of crystals in your array.	1
domainthickness	μm	Thickness of the crystal domains.	10
temperature	K	Temperature of the monochromator in Kelvin.	300
optimize		Flag to tell if the component should optimize for reflections or not.	0
x_pos	vector	x-Position of each crystal	NULL
y_pos	vector	y-Position of each crystal	NULL
z_pos	vector	z-Position of each crystal	NULL
x_rot	vector	Rotation around x-axis for each crystal	NULL
y_rot	vector	Rotation around y-axis for each crystal	NULL
z_rot	vector	Rotation around z-axis for each crystal NOTE: Rotations happen around x, then y, then z.	NULL
n_reflections		The maximum number of reflections a neutron can undergo in the crystal array.	100

Name	Unit	Description	Default
verbose		Verbosity of the monochromator. Used for debugging.	0
draw_as_rectangles		Draw the monochromators as boxes. DOES NOT WORK WHEN USING _rot parameters.	0

Links

- Component source code found in file `Monochromator_bent.comp`.
- Jan Šaroun NIM A Volume 529, Issue 1-3 (2004), pp162-165
- Christensen, D.L.; Cabeza, S.; Pirling, T.; Lefmann, K.; Šaroun, J. Simulating Neutron Diffraction from Deformed Mosaic Crystals in McStas. Quantum Beam Sci. 2026, 10, 6. https://doi.org/10.3390/qubs10010006

14.53. The `Monochromator_bent_complex` McStas Component

A bent crystal monochromator. Based on the model implemented by Jan Šaroun in NIMA 529 (2004) pp 162-165. Mosacity and bending radius can be set.

Identification

- **Author:** Daniel Lomholt Christensen <dlc@math.ku.dk> with help from Jan Šaroun
- **Origin:** ILL/NBI
- **Date:** 2 August 2025

Description

This component is a more complex implementation of Monochromator_bent. This component only differs in the fact that it allows and forces the user to set every single parameter for every single crystal in the crystal array. An exception to this rule is the plane of reflection, for which one can choose a single plane of reflection, and that will work for all crystals.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
zwidth	m	Width of each crystal without bending.	NULL
yheight	m	Height of each crystal without bending.	NULL
xthickness	m	Thickness of each crystal without bending.	NULL
radius_x	m	Radius of the circle the monochromator bends on in the plane. Can be negative.	NULL
radius_y	m	Radius of the (very large) circle the monochromator bends on as a side effect of the horizontal bending. The code assumes that it is so small that it does not affect the points of intersection appreciatively of the crystal.	NULL
angle_to_cut_horizontal	degrees	Angle between cut and normal of crystal slab, horizontally	NULL
mosaicity	arcmin	Gaussian mosaicity of the crystal. Always the horizontal mosaicity	NULL
mosaic_anisotropy	1	Anisotropy of the mosaicity, changes vertical mosaicity to be mosaic_anisotropy*mosaicity	NULL
domainthickness	μm	Thickness of the crystal domains.	NULL
temperature	K	Temperature of the monochromator in Kelvin.	NULL
plane_of_reflection	string	The plane of reflection from the material. The list of possible reflections can be seen in the source code. Each plane must be separated by a ";", like "Si400;Si111".	"Si400"
x_pos	vector	x-Position of each crystal	NULL
y_pos	vector	y-Position of each crystal	NULL
z_pos	vector	z-Position of each crystal	NULL
x_rot	vector	Rotation around x-axis for each crystal	NULL
y_rot	vector	Rotation around y-axis for each crystal	NULL
z_rot	vector	Rotation around z-axis for each crystal NOTE: Rotations happen around x, then y, then z.	NULL
n_crystals	1	Number of crystals in your array.	1
optimize	1	Flag to tell if the component should optimize for reflections or not.	0
verbose	1	Verbosity of the monochromator. Used for debugging.	0
n_reflections	1	The maximum number of reflections a neutron can undergo in the crystal array.	100

Name	Unit	Description	Default
draw_as_rectangles	1	Draw the monochromators as boxes. DOES NOT WORK WHEN USING _rot parameters.	0

Links

- Component source code found in file `Monochromator_bent_complex.comp`.
- Jan Šaroun NIM A Volume 529, Issue 1-3 (2004), pp162-165
- Christensen, D.L.; Cabeza, S.; Pirling, T.; Lefmann, K.; Šaroun, J. Simulating Neutron Diffraction from Deformed Mosaic Crystals in McStas. Quantum Beam Sci. 2026, 10, 6. https://doi.org/10.3390/qubs10010006

14.54. The `PerfectCrystal` McStas Component

Based on a perfect crystal component by: Miguel A. Gonzalez, A. Dianoux June 2013 (ILL)

Changelog: Version 1.1 - BUGFIX: correct neutron energy shift in Doppler mode - added option 'debyescherrer' to select analyzer geometry - added option 'facette' to approximate analyzer sphere by small, flat crystals

Version 1.0 - inital release

Identification

- **Author:** Markus Appel
- **Origin:** ILL / FAU Erlangen-Nuernberg
- **Date:** 2015-12-21

Description

Perfect crystal component, primarily for use as monochromator and analyzer in backscattering spectrometers. Reflection from a single Bragg reflex of a flat or spherical surface is simulated using a Darwin, Ewald or Gaussian profile. Doppler movement of the monochromator is supported as well.

Orientation of the crystal surface is *different* from other monochromator components! Gravitational energy shift for tall analyzers should work in principle, but it not tested yet. See the parameter description on how to define the geometry and properties.

[1] Website for Backscattering Spectroscopy: http://www.ill.eu/sites/BS-review/index.htm

Examples: IN16B (ILL) Si111 large angle analyzers (approximate dimensions): PerfectCrystal(radius=2, lambda=6.2708, sigma=0.24e-3, ttmin=40, ttmax=165, phimin=-45, phimax=+45, centerfocus=1)

IN16B (ILL) Si111 Doppler monochromator: PerfectCrystal(radius=2.165, lambda=6.2708, sigma=0.24e-3, width = 0.500, height = 0.250, centerfocus=0, speed = 4.7, amplitude = 0.075, exclusive=1)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
ttmin	deg	Minimum horizontal analyzer coverage angle (option 1)	NAN
ttmax	deg	analyzer coverage angle in horizontal xz-plane between -180 and 180	NAN
tt0	deg	Horizontal angular center of the analyzer coverage (option 2)	NAN
ttwidth	deg	horizontal coverage as center and full width	NAN
width	m	absolute width	NAN
phimin	deg	Minimum vertical analyzer coverage angle (option 1)	NAN
phimax	deg	vertical analyzer coverage between -90 and 90 (-180 and 180 if debyescherrer==1) option 2	NAN
phi0	deg	Vertical angular center of the analyzer coverage (option 2)	NAN
phiwidth	deg	vertical coverage as center and full width option 3	NAN
height	m	absolute height (only if debyescherrer==0)	NAN
debyescherrer	-180...180	(0/1) bend analyzer following a Debye-Scherrer ring along scattering angle tt (twotheta) (phi =	0
facette	m	width of square crystal facettes arranged on the spherical surface (set 0 to disable). Default: 0 Warning: Facettation will fail at the poles along +-y axis.	0
facette_xi	deg	random misalignemt of each facette. Default: 0	0
centerfocus	0/1	Component origin is the center of the analyzer sphere if 1, if set to 0 the origin is on the analyzer surface. Default: 0	0
radius	m	Radius of curvature, set to 0 for a flat surface. Default: 0	0

Name	Unit	Description	Default
tau	A^{-1}	Scattering vector of the reflex (sometimes also called Q...)	NAN
lambda	A	Alternatively to tau: backscattered wavelength	NAN
R0		Peak reflectivity. Default: 1	1.0
dtauovertau	1	Plateau width of the Ewald/Darwin curve (see	NAN
dtauovertau_ext		Relative variation of tau (randomized for each trajectory, full width)	0
ewald	0/1	Use Ewald curve if 1, Darwin curve if 0. Default: 1 (Ewald)	1
sigma	meV	Width of the Gaussian reflectivity curve in meV (corresponding to the energy resolution). (The width will be transformed and the Gaussian is actually calculated in k-space.)	NAN
ismirror	0/1	Simply reflect all neutrons if 1. Good for debugging/visualization. Default: 0	0
speed	m/s	Maximum Doppler speed. The actual monochromator velocity is randomized between -speed and +speed. Default: 0	0
amplitude	m	Amplitude of the Doppler movement. Default: 0	0
smartphase	0/1	Optimize Doppler phase for better MC efficiency if set to 1. *WARNING:* Experimental option! Always compare to a simulation without smartphase. Better do not use smartwidth with Ewald/Darwin curves due to their endless tails. Default: 0	0
smartwidth	meV	Half width of the possible energy reflection window for smartphase. Default: 5*sigma OR 10*2*dtauovertau*E0	NAN
exclusive	0/1	If set to 1, absorb all neutrons that missed the monochromator/analyzer surface. Default: 0	0
transmit	0...1	Monte-Carlo probability of transmitting an event through the monochromator/analyzer surface. (Events with R=1.0 for DarwinE/Ewald curves are always reflected!). Default: 0	0

Name	Unit	Description	Default
verbose	1	If non-zero, print additional information	0

Links

- Component source code found in file `PerfectCrystal.comp`.

14.55. The `Spherical_Backscattering_Analyser` McStas Component

Spherical backscattering analyser

Identification

- **Author:** Nikolaos Tsapatsaris with P Willendrup, R Lechner, H Bordallo
- **Origin:** NBI/ESS
- **Date:** 2015

Description

Spherical backscattering analyser with variable reflectivity, and mosaic gaussian response.

Based on earlier developments by Miguel A. Gonzalez 2000, Tilo Seydel 2007, Henrik Jacobsen 2011.

Example: COMPONENT doppler = Spherical_Backscattering_Analyser(xmin=0, xmax=0, ymin=0, ymax=0, mosaic=21.0, dspread=0.00035, Q=2.003886241, DM=0, radius=2.5, f_doppler=0, A_doppler=0, R0=1, debug=0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xmin	m	Minimum absolute value of x =~ 0.5 * Width of the monochromator	0
xmax	m	Maximum absolute value of x =~ 0.5 * Width of the monochromator	0
ymin	m	Minimum absolute value of x =~ 0.5 * Width of the monochromator	0
ymax	m	Maximum absolute value of y =~ 0.5 * Height of the monochromator	0
mosaic	arc min	Mosaicity, FWHM	21.0

Name	Unit	Description	Default
dspread	1	Relative d-sprea, FWHM	0.00035
Q	$Å^{-1}$	Magnitude of scattering vector	2.003886241
DM	Å	monochromator d-spacing instead of Q = 2*pi/DM	0
radius	m	Curvature radius	2.5
f_doppler	Hz	Frequency of doppler drive	0
A_doppler	m	Amplitude of doppler drive	0
R0	1	Scalar, maximum reflectivity of neutrons	1
debug	1	if debug > 0 print out some info about the calculations	0

Links

- Component source code found in file `Spherical_Backscattering_Analyser.comp`.

14.56. Contributed polarisation components

14.57. The `Foil_flipper_magnet` McStas Component

A magnet with a spin-flip foil inside

Identification

- **Author:** Erik B Knudsen
- **Origin:** DTU Physics
- **Date:** Sep. 2015

Description

ESCRIPTION A magnet with a spin-flip foil inside

This component models a magnet with a constant magnetic field and a slanted magnetized foil inside acting as a spin-flipper. This arrangement may be used to target the geometry of a Spin-Echo SANS instrument (Rekveldt, NIMB, 1997 || Rekveldt, Rev Sci Instr. 2005). In its most basic form a SE-SANS instrument nrelies on inclined field regions of constant magnetic field, which can be difficult to realize. Instead it is possible to emulate such regions using a magnetized foil.

In this component th emagntzied foil is modelled as a canted (by the angle phi) mathematical plane inside a region of constant magnetic field of dimensions (xwidth,yheight,zdepth). Furthermore, exponetially decaying stray fields from the constant field region may be specified by giving parameters (Bxwidth,Byheight,Bzdepth) > (xwidth,yheight,zdepth).

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
stray_field		Toggle for modelling of stray fields.	1
xwidth	m	width of the magnet	
yheight	m	height of the magnet	
zdepth	m	thickness of the magnet	
Bxwidth	m	width of the B-field of thev component	-1
Byheight	m	height of the B-field of the component	-1
Bzdepth	m	thickness of the B-field of the component	-1
phi	rd	angle (from the xz-plane) of the foil-spinflipper inside the magnet	0.0
foilthick	um	Thickness of the magnetized foil	0.0
Bx	T	Parameter used for x composant of field.	
By	T	Parameter used for y composant of field.	
Bz	T	Parameter used for z composant of field.	
foil_in		Toggle for optional foil in the flipper.	1
verbose		Toggle verbose mode	0

Links

- Component source code found in file `Foil_flipper_magnet.comp`.

14.58. The `Pol_bender_tapering` McStas Component

Polarising bender.

Identification

- **Author:** Erik Bergbaeck Knudsen (erkn@fysik.dtu.dk)
- **Origin:** DTU Physics
- **Date:** June 2012

Description

A component modelling a set of identical blades bent to form a multcihannel (nslits) bender. The bender has a cylindrically curved entry plane (Will be extended to also allow a flat entry plane).

The bender may also be tapered such that it can have focusing or defocusing porperties. This may be specified through the "channel_file"-parameter, which points to a data file in which the individual channels are defined by an entry- and an exit-width.

The file format for channel file columns: > #d-spacing1 d-spacing2 l_spacer d-coating1 d-coating2

```
> 1e-3  0.8e-3 1e-5  1e-9 1e-9
```

Each blades is considered coated with a reflecting coating, whose thickness is defined in the latter two columsn of the channel data file.

Coating reflectivities are read from another data file with separate reflectivities for up and down spin. When a neutron ray hits a blade the polarization vector associated with the ray is decomposed into spin-up and down probablities, which in turn determines the overall reflectivity for this ray on the mirror. Furthermore the probabilities are used to also reconstruct the polarization vector of the reflected ray.

File format for the reflectivity file: > #parameter = m > q/m R(up) R(down)

The effect of spacers inside the channels is modelled by absorption only. The depth of the spacers in a channel is considered constant for one channel (set in the channel data file), the number of them is constant across all channels and is an input parameter to the component.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
channel_file		File name of file containing channel data. such as spacer widths etc. (see above for format)	"channel.dat"
xwidth	m	Width at the bender entry. Currently unsupported.	0
yheight	m	Height at the bender entry.	
length	m	Length of blades that make up the bender.	
d_substrate	m	Thickness of the substrate.	
entry_radius	m	Radius of curvature of the entrance window.	10
radius	m	Curvature of a single plate/polarizer. +:curve left/-:right	100
G	m/s^2	Magnitude of gravitation.	9.8
nslit	1	Number of slits in the bender	1
debug		If > 0 print out some internal parameters.	1
reflect_file		File name of file containing reflectivity data parametrized either by q or m.	NULL

Name	Unit	Description	Default
sigma_abs	barn	Absorption per unit cell at v=2200 m/s of spacers. Defaults to literature value for Al.	0.231
V0	Å^3	Volume of unit cell for spacers. Defaults to Al.	66.4
nspacer		Number of spacers per channel.	5

Links

- Component source code found in file `Pol_bender_tapering.comp`.

14.59. The `Pol_triafield` McStas Component

Constant magnetic field in a isosceles triangular coil

Identification

- **Author:** Morten Sales, based on Pol_constBfield by Peter Christiansen
- **Origin:** Helmholtz-Zentrum Berlin
- **Date:** 2013

Description

Rectangular box with constant B field along y-axis (up) in a isosceles triangle. There is a guide (or precession) field as well. It is along y in the entire rectangular box. A neutron hitting outside the box opening or the box sides is absorbed.

```
------------------
|        /\       |
| Bguide/  \Bguide |      x
|      /    \      |      ^
|     /      \     |      |
|    /   B    \    |      |-----> z
|   /   and    \   |
|  /   Bguide   \  |
| /              \ |
|/________________\|
```

The angle of the inclination of the triangular field boundary is given by the arctangent to xwidth/(0.5*zdepth)

This component does NOT take gravity into account.

Example: Pol_triafield(xwidth=0.1, yheight=0.1, zdepth=0.2, B=1e-3, Bguide=0.0)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xwidth	m	Width of opening	
yheight	m	Height of opening	
zdepth	m	zdepth of field	
B	T	Magnetic field along y-direction inside triangle	0
Bguide	T	Magnetic field along y-direction inside entire box	0

Links

- Component source code found in file `Pol_triafield.comp`.

14.60. The `Pol_pi_2_rotator` McStas Component

Ideal π/2-rotator

Identification

- **Author:** Erik Knudsen (erkn@fysik.dtu.dk)
- **Origin:** Risoe
- **Date:** 8/4-2008

Description

Simple, idelized, component that turns the polarization exactly π/2 around the specified vector vector (rx,ry,rx). The geometry of the component is realized as a box, where the the spin is rotated at the z-midpoint.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xwidth	m	width of the component	0.1
yheight	m	height of the component	0.1
zdepth	m	thickness of the component	0.01
rx		x-component of the rotation axis	0

Name	Unit	Description	Default
ry		y-component of the rotation axis	0
rz		z-component of the rotation axis	1

Links

- Component source code found in file `Pol_pi_2_rotator.comp`.

14.61. The `Transmission_polarisatorABSnT` McStas Component

Transmission V-polarisator including absorption by Fe in the supermirror.

Identification

- **Author:** Andreas Ostermann
- **Origin:** TUM
- **Date:** 2004

Description

Transmission V-polarisator including absorption by Fe in the supermirror.
Example: Gravity_guide(w1=0.1, h1=0.1, w2=0.1, h2=0.1, l=12,

```
R0=0.99, Qc=0.021, alpha=6.07, m=1.0, W=0.003, k=1, d=0.0005)
```

Example: Transmission_polarisatorABSnT(w1=0.050, h1=0.050,

```
w2=0.050, h2=0.050, l=2.700,
waferD=0.0003, FeD=2.16e-06,
Si_i=0.2, Si_a=0.215,
R0=0.99, Qc=0.02174, alpha=4.25, W=0.001,
```

mleft=1.2, mright=1.2, mtop=1.2, mbottom=1.2,

```
R0_up=0.99, Qc_up=0.014, alpha_up=2.25, W_up=0.0025, mup=1.0,
```

R0_down=0.99, Qc_down=0.02174, alpha_down=3.8, W_down=0.00235, mdown=2.5)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
w1	m	Width at the polarizer entry	
h1	m	Height at the polarizer entry	
w2	m	Width at the polarizer exit	
h2	m	Height at the polarizer exit	
l	m	length of polarizer	
waferD	m	Thickness of Si wafer	
Si_i	barns	Scattering cross section per atom (barns)	
Si_a	barns	Absorption cross section per atom (barns) at 2200m/s	
FeD	m	Thickness of Fe wafer	
R0	1	Low-angle reflectivity of the outer guide	0.99
Qc	$Å^{-1}$	Critical scattering vector of the outer guide	0.02174
alpha	Å	Slope of reflectivity of the outer guide	4.25
W	$Å^{-1}$	Width of supermirror cut-off of the outer guide	0.001
R0_up	1	Low-angle reflectivity of the Fe/Si-wafer for spin up neutrons	0.99
Qc_up	$Å^{-1}$	Critical scattering vector of the Fe/Si-wafer for spin up neutrons	0.0109
alpha_up	Å	Slope of reflectivity of the Fe/Si-wafer for spin up neutrons	4.25
W_up	$Å^{-1}$	Width of supermirror cut-off of the Fe/Si-wafer for spin up neutrons	0.0025
R0_down	1	Low-angle reflectivity of the Fe/Si-wafer for spin down neutrons	0.99
Qc_down	$Å^{-1}$	Critical scattering vector of the Fe/Si-wafer for spin down neutrons	0.02174
alpha_down	Å	Slope of reflectivity of the Fe/Si-wafer for spin down neutrons	6.25
W_down	$Å^{-1}$	Width of supermirror cut-off of the Fe/Si-wafer for spin down neutrons	0.001
mleft	1	m-value of material for left. vert. mirror of the outer guide	-1
mright	1	m-value of material for right. vert. mirror of the outer guide	-1
mtop	1	m-value of material for top. horz. mirror of the outer guide	-1

Name	Unit	Description	Default
mbottom	1	m-value of material for bottom. horz. mirror of the outer guide	-1
mup	1	m-value of the Fe/Si-wafer for spin up neutrons	-1
mdown	1	m-value of the Fe/Si-wafer for spin down neutrons	-1

Links

- Component source code found in file `Transmission_polarisatorABSnT.comp`.

14.62. The `Transmission_V_polarisator` McStas Component

Transmission V-polarisator including absorption by Fe in the supermirror. Experimentally benchmarked.

Identification

- **Author:** Andreas Ostermann (additions from Michael Schneider, SNAG)
- **Origin:** TUM
- **Date:** 2024

Description

Transmission V-polarisator including absorption by Fe in the supermirror.
Example: Transmission_V_polarisator(w1=0.050, h1=0.050,

```
w2=0.050, h2=0.050, l=2.700,
waferD=0.0003, FeD=2.16e-06,
Si_i=0.2, Si_a=0.215,
R0=0.99, Qc=0.02174, alpha=4.25, W=0.001,
```

mleft=1.2, mright=1.2, mtop=1.2, mbottom=1.2, reflectUP="measured_up_q.dat",reflectDW="measured_dw

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
reflectUP	str	Reflectivity profile of the FeSi-wafer for spin-up neutrons; columns [q,R]	0

Name	Unit	Description	Default
reflectDW	str	Reflectivity profile of the FeSi-wafer for spin-down neutrons; columns [q,R]	0
w1	m	Width at the polarizer entry	
h1	m	Height at the polarizer entry	
w2	m	Width at the polarizer exit	
h2	m	Height at the polarizer exit	
l	m	length of polarizer	
waferD	m	Thickness of Si wafer	
Si_i	barns	Scattering cross section per atom (barns)	
Si_a	barns	Absorption cross section per atom (barns) at 2200m/s	
FeD	m	Thickness of Fe in supermirror, Ti is neglected	
R0	1	Low-angle reflectivity of the outer guide	0.99
Qc	$Å^{-1}$	Critical scattering vector of the outer guide	0.02174
alpha	Å	Slope of reflectivity of the outer guide	4.25
W	$Å^{-1}$	Width of supermirror cut-off of the outer guide	0.001
mleft	1	m-value of material for left. vert. mirror of the outer guide	-1
mright	1	m-value of material for right. vert. mirror of the outer guide	-1
mtop	1	m-value of material for top. horz. mirror of the outer guide	-1
mbottom	1	m-value of material for bottom. horz. mirror of the outer guide	-1

Links

- Component source code found in file `Transmission_V_polarisator.comp`.
- P. Böni, W. Münzer and A. Ostermann: Physica B: Condensed Matter Volume 404, Issue 17, 1 September 2009, Pages 2620-2623

14.63. The `Spin_random` McStas Component

Set a random polarisation

Identification

- **Author:** Michael Schneider (SNAG)

- **Origin:** SNAG
- **Date:** 2024

Description

This component has no physical size or extent, it simply asigns the neutron ray polarisation randomly to either spin-up or spin-down.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default

Links

- Component source code found in file `Spin_random.comp`.

14.64. The `PSD_Pol_monitor` McStas Component

Position-sensitive monitor.

Identification

- **Author:** Alexander Backs, based on PSD_monitor by K. Lefmann
- **Origin:** ESS
- **Date:** 2022

Description

An (n times m) pixel PSD monitor, measuring local polarisation as function of x,y coordinates.

Example: PSD_Pol_monitor(xmin=-0.1, xmax=0.1, ymin=-0.1, ymax=0.1, nx=90, ny=90, my=1, filename="Output.psd")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nx	1	Number of pixel columns	90
ny	1	Number of pixel rows	90

Name	Unit	Description	Default
filename	string	Name of file in which to store the detector image	0
xmin	m	Lower x bound of detector opening	-0.05
xmax	m	Upper x bound of detector opening	0.05
ymin	m	Lower y bound of detector opening	-0.05
ymax	m	Upper y bound of detector opening	0.05
xwidth	m	Width of detector. Overrides xmin, xmax	0
yheight	m	Height of detector. Overrides ymin, ymax	0
restore_neutron	1	If set, the monitor does not influence the neutron state	0
nowritefile	1	If set, monitor will skip writing to disk	0
mx	1	Define the projection axis along which the polarizatoin is evaluated, x-component	0
my	1	Define the projection axis along which the polarizatoin is evaluated, y-component	0
mz	1	Define the projection axis along which the polarizatoin is evaluated, z-component	0

Links

- Component source code found in file `PSD_Pol_monitor.comp`.

14.65. The `PSD_spinDmon` McStas Component

Position-sensitive monitor, measuring the spin-down component of the neutron beam.

Identification

- **Author:** Michael Schneider (SNAG)
- **Origin:** SNAG
- **Date:** 2024

Description

An (n times m) pixel PSD monitor, measuring the spin-down component of the neutron beam.

Example: PSD_monitor(xmin=-0.1, xmax=0.1, ymin=-0.1, ymax=0.1, nx=90, ny=90, filename="Output.psd")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nx	1	Number of pixel columns	90
ny	1	Number of pixel rows	90
filename	string	Name of file in which to store the detector image	0
xmin	m	Lower x bound of detector opening	-0.05
xmax	m	Upper x bound of detector opening	0.05
ymin	m	Lower y bound of detector opening	-0.05
ymax	m	Upper y bound of detector opening	0.05
xwidth	m	Width of detector. Overrides xmin, xmax	0
yheight	m	Height of detector. Overrides ymin, ymax	0
restore_neutron	1	If set, the monitor does not influence the neutron state	0
nowritefile	1	If set, monitor will skip writing to disk	0

Links

- Component source code found in file `PSD_spinDmon.comp`.

14.66. The `PSD_spinUmon` McStas Component

Position-sensitive monitor, measuring the spin-up component of the neutron beam.

Identification

- **Author:** Michael Schneider (SNAG)
- **Origin:** SNAG
- **Date:** 2024

Description

An (n times m) pixel PSD monitor, measuring the spin-up component of the neutron beam.

Example: PSD_monitor(xmin=-0.1, xmax=0.1, ymin=-0.1, ymax=0.1, nx=90, ny=90, filename="Output.psd")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nx	1	Number of pixel columns	90
ny	1	Number of pixel rows	90
filename	string	Name of file in which to store the detector image	0
xmin	m	Lower x bound of detector opening	-0.05
xmax	m	Upper x bound of detector opening	0.05
ymin	m	Lower y bound of detector opening	-0.05
ymax	m	Upper y bound of detector opening	0.05
xwidth	m	Width of detector. Overrides xmin, xmax	0
yheight	m	Height of detector. Overrides ymin, ymax	0
restore_neutron	1	If set, the monitor does not influence the neutron state	0
nowritefile	1	If set, monitor will skip writing to disk	0

Links

- Component source code found in file `PSD_spinUmon.comp`.

14.67. Contributed single-crystal and powder/polycrystalline samples

14.68. The `Single_crystal_inelastic` McStas Component

An extension of Single crystal with material dispersion

Identification

- **Author:** Duc Le
- **Origin:** STFC
- **Date:** August 2020

Description

The component handles a 4D S(q,w) material dispersion, and scatters neutrons out of it. It is based on the Isotropic_Sqw methodology, extended to 4D.

A number of approximations are applied:

- geometry is restricted to box, cylinder and sphere - ignore absorption, multiple scattering and incoherent scattering - no temperature handling. The temperature must be taken into account when generating the S(q,w) data sets, which should contain Bose/detailed balance. - intensity is used as provided by the S(q,w) data set

We recommend that you first try the Test_Single_crystal_inelastic example to learn how to use this complex component.

4D S(q,w) data files

```
--------------------
```

The input data file derives from other McStas formats. It may contain structural information as in Lau/Laz files, and a list of S(q,w) [one per row] with 5 columns

```
[ H K L E S(q,w) ]
```

An example file example.sqw4 is provided in the McStas/Data directory.

You may generate such files using iFit such as:

```
s = sqw_vaks('defaults');              % a 4D model
d = iData(s);                          % use default coarse grid for axes [-0.5:0.5]
saveas(d, 'sx_coh.sqw4', 'mcstas');    % export to 4D Sqw file
```

%VALIDATION: This component is undergoing validation.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
mosaic_AB	arcmin,arcmin,1,1,1,1	In-plane mosaic rotation and plane vectors (anisotropic), mosaic_A, mosaic_B, A_h,A_k,A_l, B_h,B_k,B_l. Puts the crystal in the in-plane mosaic state. Vectors A and B define plane in which the crystal roation is defined, and mosaic_A, mosaic_B, denotes the resp. mosaicities (gaussian RMS) with respect to the the two reflections chosen by A and B (Miller indices).	{0,0, 0,0,0, 0,0,0}

Name	Unit	Description	Default
sqw	string	Name of a 4d S(q,w) file - example.sqw4 is included in your installation	0
geometry	string	Name of an OFF, PLY or STL file for complex geometry. An OFF file may be generated from XYZ coordinates using qhull/powercrust. The mesh must be a single, closed object: only the first entry and exit along the neutron path are used, so a second part of the mesh further along the path (disjoint parts, or a concave shape the path re-enters) is not seen.	0
xwidth	m	Width of crystal.	0
yheight	m	Height of crystal.	0
zdepth	m	Depth of crystal (no extinction simulated)	0
radius	m	Outer radius of sample in (x,z) plane.	0
delta_d_d	1	Lattice spacing variance, gaussian RMS.	1e-4
mosaic	arcmin	Isotropic crystal mosaic, gaussian RMS. Puts the crystal in the isotropic mosaic model state, thus disregarding other mosaicity parameters.	-1
mosaic_a	arcmin	Horizontal (rotation around lattice vector a) mosaic (anisotropic), gaussian RMS. Put the crystal in the anisotropic crystal vector state. I.e. model mosaicity through rotation around the crystal lattice vectors. Has precedence over in-plane mosaic model.	-1
mosaic_b	arcmin	Vertical (rotation around lattice vector b) mosaic (anisotropic), gaussian RMS.	-1
mosaic_c	arcmin	Out-of-plane (Rotation around lattice vector c) mosaic (anisotropic), gaussian RMS.	-1
recip_cell	1	Choice of direct/reciprocal (0/1) unit cell definition.	0
barns	1	Flag to indicate if |F|^2 from 'reflections' is in barns or fm^2. barns=1 for laz and isotropic constant elastic scattering (reflections=NULL), barns=0 for lau type files.	0

Name	Unit	Description	Default
ax	Å / $Å^{-1}$	Coordinates of first (direct/recip) unit cell vector.	0
ay	Å / $Å^{-1}$	a on y axis	0
az	Å / $Å^{-1}$	a on z axis	0
bx	Å / $Å^{-1}$	Coordinates of second (direct/recip) unit cell vector.	0
by	Å / $Å^{-1}$	b on y axis	0
bz	Å / $Å^{-1}$	b on z axis	0
cx	Å / $Å^{-1}$	Coordinates of third (direct/recip) unit cell vector.	0
cy	Å / $Å^{-1}$	c on y axis	0
cz	Å / $Å^{-1}$	c on z axis	0
p_transmit	1	Monte Carlo probability for neutrons to be transmitted without any scattering. Used to improve statistics from weak reflections.	-1
sigma_abs	barns	Absorption cross-section per unit cell at 2200 m/s.	0
sigma_inc	barns	Incoherent scattering cross-section per unit cell.	0
aa	deg	Unit cell angles alpha, beta and gamma. Then uses norms of vectors a,b and c as lattice parameters.	0
bb	deg	Beta angle.	0
cc	deg	Gamma angle.	0
order	1	Limit multiple scattering up to given order (0: all, 1: first, 2: second, ...).	0
RX	m	Radius of horizontal along X lattice curvature. flat for 0.	0
RY	m	Radius of vertical lattice curvature. flat for 0.	0

Links

- Component source code found in file `Single_crystal_inelastic.comp`.

14.69. The `SiC` McStas Component

SiC layer sample

Identification

- **Author:** S. Rycroft
- **Origin:** IRI.
- **Date:** Jan 2000

Description

SiC layer sample
Example: SiC()

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xlength	m	length of SiC plate	
yheight	m	height of SiC plate	

Links

- Component source code found in file `SiC.comp`.

14.70. The `Spot_sample` McStas Component

Spot sample.

Identification

- **Author:** Garrett Granroth
- **Origin:** Oak Ridge National Laboratory
- **Date:** 14.11.14

Description

A sample that is a series of delta functions in omega and Q. The Q is determined by a two theta value and an equal number of spots around the beam center. This component is a variation of the V sample written by Kim Lefmann and Kristian Nielsen. The following text comes from their original component. A Double-cylinder shaped incoherent scatterer (a V-sample) No multiple scattering. Absorbtion included. The shape of the sample may be a box with dimensions xwidth, yheight, zthick. The area to scatter to is a disk of radius 'focus_r' situated at the target. This target area may also be rectangular if specified

focus_xw and focus_yh or focus_aw and focus_ah, respectively in meters and degrees. The target itself is either situated according to given coordinates (x,y,z), or setting the relative target_index of the component to focus at (next is +1). This target position will be set to its AT position. When targeting to centered components, such as spheres or cylinders, define an Arm component where to focus at.

Example: spot_sample(radius_o=0.01, h=0.05, pack = 1, xwidth=0, yheight=0, zthick=0, Eideal=100.0,w=50.0,two_theta=25.0,n_spots=4)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius_o	m	Outer radius of sample in (x,z) plane	0.01
h	m	Height of sample y direction	0.05
pack	1	Packing factor	1
xwidth	m	horiz. dimension of sample, as a width	0
yheight	m	vert.. dimension of sample, as a height	0
zthick	m	thickness of sample	0
Eideal	meV	The presumed incident energy	100.0
w	meV	The energy transfer of the delta function	50.0
two_theta	degrees	the scattering angle of the spot	25.0
n_spots	1	The number of spots to generate symmetrically around the beam	4

Links

- Component source code found in file `Spot_sample.comp`.

14.71. The `GISANS_sample` McStas Component

Sample for GISANS

Identification

- **Author:** H. Frielinghaus
- **Origin:** FZJ
- **Date:** March 2023

Description

The idea of this sample goes back to the real sample studied in the publication https://doi.org/10.1063/
This is a sandwich of a sapphire and silicon slab through which the neutrons impinge and the sample in between where the scattering emerges from. The sample consist of spherical polystyrene colloids in heavy water that form aligned crystallites at the solid-liquid interface, while in the middle of the volume the crytallites are orientationally averaged in the polar angle that points in the lateral direaction. The crystallites of the current verision are not highly realistic because they are monoclinic with a 60° angle mimicking a fraction of a hexagonal structure with hexagonal planes stacked in the z-direction. Inside the routine one can change between 2 or 3 different planes. Also the total crystallite size can be changed. Especially the parallelogram with 60° angle is unrealistic because one usually would assume the crystallites to have a hexagonal shape. However, the finite crystallites mimic the final correlation length inside the sample. The sample works for neutrons impingin from the sapphire, the silicon at grazing angles and in transmission in the sense of a SANS sample. If the neutrons impinge through the other layers (sample and silicon oxide) - the responses are not so realistic and need further development. Also multiple scattering would be needed to be implemented still - but could be done.

What this sample component covers relatively well is the fact that damping of the intensity by scattering and absorption is also covered. That means that even at larger angles than the critical angle the intensity is damped inside the sample and so only a certain near surface layer scatters neutrons. This effect is not covered by BornAgain.

I see this as a start for further developments.

Information extracted from malformatted docstrings follow:

the total thickness of the sandwich is zsapph+zsamp+zsilicon the thicknesses zsampsurf and zsiliconsurf are parts of the total sample/silicon thicknesses

(scattering length densitites, reciprocal total cross section of absorption, reciprocal total cross section of incoherent scattering) units rho... [A-2], abslen... [cm], inclen... [cm]

```
rhosapph,       abslensapph,       inclensapph        for sapphire
rhoD2O,         abslenD2O,         inclenD2O          for D2O as solvent in sample
rhoPS,          abslenPS,          inclenPS           for polystyrene as colloids in
```

rhosiliconsurf, abslensiliconsurf, inclensiliconsurf for silicon surface (oxide layer)

```
rhosilicon,     abslensilicon,     inclensilicon      for silicon
```

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xwidth	m	Width of sample volume	0.05

Name	Unit	Description	Default
yheight	m	Height of sample volume	0.15
zsapph	m	Thickness of the sapphire layer on one side	0.02
zsamp	m	Thickness of the overall sample between sapphire and silicon	0.002
zsampsurf	m	Thickness from the total of zsamp on either side of sapphire/silicon that is oriented	0.000001
zsilicon	m	Thickness of the silicon layer on the other side	0.02
zsiliconsurf	m	Thickness from the total of zsilicon on the sample side (oxidization layer)	5e-9
rhosapph	$Å^{-2}$	Scattering length density of sapphire	5.773e-6
abslensapph	cm	Absorption length of sapphire for 1 Å neutrons (scales with wavelength)	163.708
inclensapph	cm	Incoherent scattering attenuation length of sapphire	49.815
rhoD2O	$Å^{-2}$	Scattering length density of D2O	6.364e-6
abslenD2O	cm	Absorption length of D2O for 1 Å neutrons (scales with wavelength)	44066.347
inclenD2O	cm	Incoherent scattering attenuation length of D2O	7.258
rhoPS	$Å^{-2}$	Scattering length density of the polystyrene colloids	1.358e-6
abslenPS	cm	Absorption length of the polystyrene colloids for 1 Å neutrons (scales with wavelength)	114.502
inclenPS	cm	Incoherent scattering attenuation length of the polystyrene colloids	0.24588
rhosiliconsurf	$Å^{-2}$	Scattering length density of the silicon surface (oxidation) layer	4.123e-6
abslensiliconsurf	cm	Absorption length of the silicon surface (oxide) layer for 1 Å neutrons (scales with wavelength)	401.051
inclensiliconsurf	cm	Incoherent scattering attenuation length of the silicon surface (oxide) layer	169.11
rhosilicon	$Å^{-2}$	Scattering length density of silicon	2.079e-6
abslensilicon	cm	Absorption length of silicon for 1 Å neutrons (scales with wavelength)	209.919
inclensilicon	cm	Incoherent scattering attenuation length of silicon	9901.6
phiPS	1	Concentration vol/vol of colloids in D2O	0.1

Name	Unit	Description	Default
Rad	A	Radius of colloids in Aangstroem	370.0
phirot	rad	angle of the ordered (aligned) crytallites at the two surfaces towards sapphire/silicon	0.0
sc_aim	1	how much is scattered versus transmitted	0.98
sans_aim	1	how much goes to the small angle range versus wide angles	0.98

Links

- Component source code found in file `GISANS_sample.comp`.

14.72. Contributed inelastic (phonon and magnon) samples

14.73. The `Dispersion_relation` McStas Component

A sample for magnon or phonon scattering based on numerical cross sections

Identification

- **Author:** Daniel Lomholt Christensen and Kim Lefmann
- **Origin:** NBI
- **Date:** 22.06.23

Description

Single-cylinder shape. Absorption included. No multiple scattering. No incoherent scattering emitted. No attenuation from coherent scattering. No Bragg scattering. Any crystal system: the unit cell is given either by its lengths and angles (a, b, c, aa, bb, cc) or by its real-space lattice vectors (ax ... cz).

Algorithm: 0. Always perform the scattering if possible (otherwise ABSORB) 1. Choose a dispersion mode, a scattering point and a direction within a focusing solid angle 2. Calculate the zeros of (E_i-E_f-hbar omega(kappa)) as a function of k_f 3. Choose one value of k_f (always at least one is possible!) 4. Perform the correct weight transformation

Dispersion files: folder_path holds one file per dispersion mode. Each file has the columns

```
h  k  l  E  I
```

on a regular grid covering one unit cell in reciprocal lattice units: h, k, l are the coordinates of q along the reciprocal lattice vectors a*, b*, c*, h = a.q/(2 pi) and so on. For a cell with 90 degree angles and no vectors given, a*, b*, c* lie along the sample x, y, z axes, as a, b, c do in SpinWave_BCO and Phonon_simple. E [meV] is the mode energy and I [barn/sr per unit cell of the given lattice] is the partial differential scattering cross section of the mode, as the factor in front of the delta function: d2sigma/dOmega dE_f = (k_f/k_i) * DW * I(q) * n_B * delta(hbar omega - E(q)). The component adds k_f/k_i, the Debye-Waller factor DW and the Bose factor n_B = n(omega) + 1 for neutron energy loss and n(omega) for energy gain, so I should not include them. I is looked up at the scattering vector folded into the unit cell.

Weight: p *= exp(-mu l) * dOmega * l_full * m * n_modes * rho_cell

```
* (k_f/k_i) * DW * I(q) * n_B * |dE_f/d(E(q) - |E_i - E_f|)|
```

with l the path length in the sample, dOmega the focusing solid angle, l_full the path length without scattering, m the number of final velocities found, n_modes the number of dispersion files and rho_cell the density of unit cells.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	m	Outer radius of sample in (x,z) plane.	
yheight	m	Height of sample in y direction.	
sigma_abs	barns	Absorption cross section at 2200 m/s per unit cell.	
sigma_inc	barns	Incoherent scattering cross section per unit cell.	
a	Å	Lattice constant, length of the a vector.	0
b	Å	Lattice constant, length of the b vector.	0
c	Å	Lattice constant, length of the c vector.	0
DW	1	Debye-Waller factor.	
T	K	Temperature.	
aa	deg	Angle alpha between b and c.	90
bb	deg	Angle beta between a and c.	90
cc	deg	Angle gamma between a and b with lengths and angles, a lies along x, b in the x-y plane and c completes a right handed set. With 90 degree angles, a, b, c lie along x, y, z.	90

Name	Unit	Description	Default
ax	Å	x component of the real-space lattice vector a. If any of ax ... cz is non-zero, the vectors are used as given in the component frame and a, b, c, aa, bb, cc are ignored.	0
ay	Å	y component of a.	0
az	Å	z component of a.	0
bx	Å	x component of b.	0
by	Å	y component of b.	0
bz	Å	z component of b.	0
cx	Å	x component of c.	0
cy	Å	y component of c.	0
cz	Å	z component of c.	0
folder_path		Folder with one dispersion file per mode, see above.	
target_x	m	position of target to focus at . Transverse coordinate.	0
target_y	m	position of target to focus at. Vertical coordinate.	0
target_z	m	position of target to focus at. Straight ahead.	0
target_index	1	relative index of component to focus at, e.g. next is +1.	0
focus_r	m	Radius of sphere containing target.	0
focus_xw	m	horiz. dimension of a rectangular area.	0
focus_yh	m	vert. dimension of a rectangular area.	0
focus_aw	deg	horiz. angular dimension of a rectangular area.	0
focus_ah	deg	vert. angular dimension of a rectangular area.	0
e_steps_low	1	Number of intervals the final velocity is searched in for roots on the energy loss side of the elastic line.	100
e_steps_high	1	Number of intervals on the energy gain side. Each interval holds at most one root, so two roots closer than an interval are missed: increase these for steep or crossing modes.	100
verbose	1	Prints information about the component if set to 1.	0

Links

- Component source code found in file `Dispersion_relation.comp`.
- The test instrument Test_Dispersion_relation_TAS.instr.

14.74. The `Phonon_BvK_PG` McStas Component

A sample for phonon scattering based on cross section expressions from the Boothroyd textbook based upon the algorithm from component Phonon_simple (with expressions from Squires, Ch. 3) Using the Born-von Karman force constants and the normal mode formalism

Identification

- **Author:** Kim Lefmann
- **Origin:** NBI, KU
- **Date:** 06.12.2025

Description

Single cylinder or slab shaped sample. Absorption included. No multiple scattering. No incoherent scattering emitted, but incoherent is present in attenuation No attenuation from coherent scattering. No Bragg scattering. Specialized for Pyrolytic Graphite (PG)

Algorithm: 0. Always perform the scattering if possible (otherwise ABSORB) 1. Choose direction within a focusing solid angle 2. Select a phonon mode at random (alternatively mode number fixed by user) 3. Calculate the zeros of (E_i-E_f-hbar omega(kappa)) as a function of k_f 4. Choose one value of k_f (there always at least one possible; see e.g. Squires) 5. Perform the correct weight transformation For details: see article A. F. Davidsen et al, manuscript for J Neutron Res 2026

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
hh	rlu	h-coordinate (in units of a*) of the q-point used when calculating the dispersion	0
kk	rlu	k-coordinate (in units of a*) of the q-point used when calculating the dispersion	0

Name	Unit	Description	Default
ll	rlu	l-coordinate (in units of c*) of the q-point used when calculating the dispersion	0
radius	m	Outer radius of sample in (x,z) plane	0
xwidth	m	Horiz. dimension of sample if slab	0
yheight	m	Height of sample in y direction	0
zdepth	m	Depth of sample if slab	0
sigma_abs	barns	Absorption cross section at 2200 m/s per atom	0
sigma_inc	barns	Incoherent scattering cross section per atom	0
DW	1	Debye-Waller factor; scalar value (no q-dependence)	1
T	K	Temperature	2
focus_r	m	Radius of sphere containing target.	0
focus_xw	m	horiz. dimension of a rectangular area	0
focus_yh	m	vert. dimension of a rectangular area	0
focus_aw	deg	horiz. angular dimension of a rectangular area	0
focus_ah	deg	vert. angular dimension of a rectangular area	0
target_x	m	position of target to focus at . Transverse coordinate	0
target_y	m	position of target to focus at. Vertical coordinate	0
target_z	m	position of target to focus at. Straight ahead.	0
target_index	1	relative index of component to focus at, e.g. next is +1	0
mode_input	1	order of the phonon mode to be scattered (unphysical). In the range 0 through 11 . If mode_input == 12, then the mode is chosen at random	12
e_steps_low	1	Amount of possible intersections beneath the elastic line	100
e_steps_high	1	Amount of possible intersections above the elastic line	100
verbose_input	1	toggles verbose mode (verbose>1) or eigenvector mode (verbose==1)	0
dispersion	1	0: do nothing special; 1: calculate and output the dispersion; 2: calculate and output the dispersion while varying v_f	0

Links

- Component source code found in file `Phonon_BvK_PG.comp`.
- The test/example instrument Test_Phonon_BvK_PG.instr.

14.75. The `SpinWave_BCO` McStas Component

A sample for FM and AFM magnon scattering from a oI lattice based on cross section expressions from Marshall and Lovesey ch.9.

Identification

- **Author:** Silas Schack (algorithm based on "Phonon_simple" by Kim Lefmann)
- **Origin:** NBI
- **Date:** 13.06.25

Description

Single-cylinder shape. Absorption included. No multiple scattering. No incoherent scattering emitted. No attenuation from coherent scattering. No Bragg scattering. Bravais lattice only. (i.e. just one spin per sublattice unit cell)

Algorithm: 0. Always perform the scattering if possible (otherwise ABSORB) 1. Choose direction within a focusing solid angle 2. Choose mode seperated by applied B-field 3. Calculate the zeros of (E_i-E_f-hbar omega(kappa)) as a function of k_f 4. Choose one value of k_f (always at least one is possible!) 5. Perform the correct weight transformation

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	m	Outer radius of sample in (x,z) plane	
yheight	m	Height of sample in y direction	
sigma_abs	barns	Absorption cross section at 2200 m/s per atom	1
sigma_inc	barns	Incoherent scattering cross section per atom	0
a1	Å	Lattice constants of orthorhombic body centred lattice	
a2	Å	Lattice constants of orthorhombic body centred lattice	

Name	Unit	Description	Default
a3	Å	Lattice constants of orthorhombic body centred lattice	
j	meV	Coupling constant to body centered spins	
ja	meV	Coupling constant to nearest neighbours along a	
jb	meV	Coupling constant to nearest neighbours along b	
jc	meV	Coupling constant to nearest neighbours along c	
j_110	meV	Coupling constant to nearest neighbours +x+y	0
j_110_prime	meV	Coupling constant to nearest neighbours +x-y	0
S	1	Spin of magnetic ions	
D	meV	Uniaxial anisotropy; easy axis is c-axis. Non-positive value assumed	
B	T	Field strength of external magnetic field applied along z-axis	
FM	1	Tag for ferromagnetic system if FM==1	0
Fq	1	Magnetic form factor	1
mode_input	1	Index of mode(s) to simulate	2
DW	1	Debye-Waller factor	1
T	K	Temperature	
e_steps_low	1	Number of low-energy steps in zrid	
e_steps_high	1	Number of high-energy steps in zrid	
target_x	m	Position of target to focus at. Transverse coordinate	0
target_y	m	Position of target to focus at. Vertical coordinate	0
target_z	m	Position of target to focus at. Straight ahead.	0
target_index	1	Relative index of component to focus at, e.g. next is +1	0
focus_r	m	Radius of sphere containing target.	0
focus_xw	m	Horiz. dimension of a rectangular area	0
focus_yh	m	Vert. dimension of a rectangular area	0
focus_aw	deg	Horiz. angular dimension of a rectangular area	0
focus_ah	deg	Vert. angular dimension of a rectangular area	0
verbose	1	Verbosity level	0

Links

- Component source code found in file `SpinWave_BCO.comp`.

14.76. Contributed SANS samples

14.77. The `SANSCurve` McStas Component

A component mimicking the scattering from a given I(q)-curve by using linear interpolation between the given points.

Identification

- **Author:** Søren Kynde (kynde@nbi.dk)
- **Origin:** KU-Science
- **Date:** October 17, 2012

Description

A box-shaped component simulating the scattering from any given I(q)-curve. The component uses linear interpolation to generate the points necessary to compute the scattering of any given neutron.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
DeltaRho	cm/Å^3	Excess scattering length density of the particles.	1.0e-14
Volume	Å^3	Volume of the particles.	1000.0
Concentration	mM	Concentration of sample.	0.01
AbsorptionCrosssection	1/m	Absorption cross section of the sample.	0.0
xwidth	m	Dimension of component in the x-direction.	
yheight	m	Dimension of component in the y-direction.	
zdepth	m	Dimension of component in the z-direction.	
SampleToDetectorDistance		Distance from sample to detector (for focusing the scattered neutrons).	
DetectorRadius	m	Radius of the detector (for focusing the scattered neutrons).	

Name	Unit	Description	Default
FileWithCurve		Datafile with the given I(q).	"Curve.dat"

Links

- Component source code found in file `SANSCurve.comp`.

14.78. The `SANSCylinders` McStas Component

A sample of monodisperse cylindrical particles in solution.

Identification

- **Author:** Martin Cramer Pedersen (mcpe@nbi.dk)
- **Origin:** KU-Science
- **Date:** October 17, 2012

Description

A component simulating the scattering from a box-shaped, thin solution of monodisperse, cylindrical particles.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
R	Å	Semiaxis of the cross section of the cylinder.	40.0
Height	Å	Height of the cylinder.	100.0
Concentration	mM	Concentration of sample.	0.01
DeltaRho	cm/Å^3	Excess scattering length density of the particles.	1.0e-14
AbsorptionCrosssection	1/m	Absorption cross section of the sample.	0.0
xwidth	m	Dimension of component in the x-direction.	
yheight	m	Dimension of component in the y-direction.	
zdepth	m	Dimension of component in the z-direction.	
SampleToDetectorDistance	m	Distance from sample to detector (for focusing the scattered neutrons).	

Name	Unit	Description	Default
DetectorRadius	m	Radius of the detector (for focusing the scattered neutrons).	

Links

- Component source code found in file `SANSCylinders.comp`.

14.79. The `SANSEllipticCylinders` McStas Component

A sample of monodisperse cylindrical particles with elliptic cross section in solution.

Identification

- **Author:** Martin Cramer Pedersen (mcpe@nbi.dk)
- **Origin:** KU-Science
- **Date:** October 17, 2012

Description

A component simulating the scattering from a box-shaped, thin solution of monodisperse, cylindrical particles with elliptic cross section.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
R1	Å	First semiaxis of the cross section of the elliptic cylinder.	20.0
R2	Å	Second semiaxis of the cross section of the elliptic cylinder.	40.0
Height	Å	Height of the elliptic cylinder.	100.0
Concentration	mM	Concentration of sample.	0.01
DeltaRho	cm/Å^3	Excess scattering length density of the particles.	1.0e-14
AbsorptionCrosssection	1/m	Absorption cross section of the sample.	0.0
xwidth	m	Dimension of component in the x-direction.	
yheight	m	Dimension of component in the y-direction.	

Name	Unit	Description	Default
zdepth	m	Dimension of component in the z-direction.	
SampleToDetectorDistance	m	Distance from sample to detector (for focusing the scattered neutrons).	
DetectorRadius	m	Radius of the detector (for focusing the scattered neutrons).	

Links

- Component source code found in file `SANSEllipticCylinders.comp`.

14.80. The `SANSLiposomes` McStas Component

A sample of polydisperse liposomes in solution (water).

Identification

- **Author:** Martin Cramer Pedersen (mcpe@nbi.dk)
- **Origin:** KU-Science
- **Date:** October 17, 2012

Description

A component simulating the scattering from a box-shaped, thin solution (water) of liposomes described by a pentuple-shell model.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
Radius	Å	Average thickness of the liposomes.	800.0
Thickness	Å	Thickness of the bilayer.	38.89
SigmaRadius		Relative Gaussian deviation of the radius in the distribution of liposomes.	0.20
VolumeOfHeadgroup	$Å^3$	Volume of one lipid headgroup - default is POPC.	319.0
VolumeOfCH2Tail	$Å^3$	Volume of the CH2-chains of one lipid - default is POPC.	818.8
VolumeOfCH3Tail	$Å^3$	Volume of the CH3-tails of one lipid - default is POPC.	108.6

Name	Unit	Description	Default
ScatteringLengthOfHeadgroup	cm	Scattering length of one lipid headgroup - default is POPC in D2O.	7.05E-12
ScatteringLengthOfCH2Tail	cm	Scattering length of the CH2-chains of one lipid - default is POPC in D2O.	-1.76E-12
ScatteringLengthOfCH3Tail	cm	Scattering length of the CH3-tails of one lipid - default is POPC in D2O.	-9.15E-13
Concentration	mM	Concentration of sample.	0.01
RhoSolvent	cm/Å^3	Scattering length density of solvent - default is D2O.	6.4e-14
AbsorptionCrosssection	1/m	Absorption cross section of the sample.	0.0
xwidth	m	Dimension of component in the x-direction.	
yheight	m	Dimension of component in the y-direction.	
zdepth	m	Dimension of component in the z-direction.	
SampleToDetectorDistance	m	Distance from sample to detector (for focusing the scattered neutrons).	
DetectorRadius	m	Radius of the detector (for focusing the scattered neutrons).	

Links

- Component source code found in file `SANSLiposomes.comp`.

14.81. The `SANSNanodiscs` McStas Component

A sample of monodisperse phospholipid bilayer nanodiscs in solution (water).

Identification

- **Author:** Martin Cramer Pedersen (mcpe@nbi.dk)
- **Origin:** KU-Science
- **Date:** October 17, 2012

Description

A component simulating the scattering from a box-shaped, thin solution (water) of monodisperse phospholipid bilayer nanodiscs.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
AxisRatio		Axis ratio of the bilayer patch.	1.4
NumberOfLipids		Number of lipids per nanodisc.	130.0
AreaPerLipidHeadgroup	$Å^2$	Area per lipid headgroup - default is POPC.	65.0
HeightOfMSP	Å	Height of the belt protein - default is MSP1D1.	24.0
VolumeOfOneMSP	$Å^3$	Volume of one belt protein - default is MSP1D1.	26296.5
VolumeOfHeadgroup	$Å^3$	Volume of one lipid headgroup - default is POPC.	319.0
VolumeOfCH2Tail	$Å^3$	Volume of the CH2-chains of one lipid - default is POPC.	818.8
VolumeOfCH3Tail	$Å^3$	Volume of the CH3-tails of one lipid - default is POPC.	108.6
ScatteringLengthOfOneMSP	cm	Scattering length of one belt protein - default is MSP1D1 in D2O.	8.8E-10
ScatteringLengthOfHeadgroup	cm	Scattering length of one lipid headgroup - default is POPC in D2O.	7.05E-12
ScatteringLengthOfCH2Tail	cm	Scattering length of the CH2-chains of one lipid - default is POPC in D2O.	-1.76E-12
ScatteringLengthOfCH3Tail	cm	Scattering length of the CH3-tails of one lipid - default is POPC in D2O.	-9.15E-13
Roughness		Factor used to smear the interfaces of the nanodisc.	5.0
Concentration	mM	Concentration of sample.	0.01
RhoSolvent	$cm/Å^3$	Scattering length density of solvent - default is D2O.	6.4e-14
AbsorptionCrosssection	1/m	Absorption cross section of the sample.	0.0
xwidth	m	Dimension of component in the x-direction.	
yheight	m	Dimension of component in the y-direction.	
zdepth	m	Dimension of component in the z-direction.	
SampleToDetectorDistance	m	Distance from sample to detector (for focusing the scattered neutrons).	
DetectorRadius	m	Radius of the detector (for focusing the scattered neutrons).	

Links

- Component source code found in file `SANSNanodiscs.comp`.

14.82. The `SANSNanodiscsFast` McStas Component

A sample of monodisperse phospholipid bilayer nanodiscs in solution (water).

Identification

- **Author:** Martin Cramer Pedersen (mcpe@nbi.dk)
- **Origin:** KU-Science
- **Date:** October 17, 2012

Description

A component very similar to EmptyNanodiscs.comp - however, the scattering profile is only computed once, and linear interpolation is the used to simulate the instrument.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
NumberOfQBins		Number of points generated in inital scattering profile.	200
AxisRatio		Axis ratio of the bilayer patch.	1.4
NumberOfLipids		Number of lipids per nanodisc.	130.0
AreaPerLipidHeadgroup	$Å^2$	Area per lipid headgroup - default is POPC.	65.0
HeightOfMSP	Å	Height of the belt protein - default is MSP1D1.	24.0
VolumeOfOneMSP	$Å^3$	Volume of one belt protein - default is MSP1D1.	26296.5
VolumeOfHeadgroup	$Å^3$	Volume of one lipid headgroup - default is POPC.	319.0
VolumeOfCH2Tail	$Å^3$	Volume of the CH2-chains of one lipid - default is POPC.	818.8
VolumeOfCH3Tail	$Å^3$	Volume of the CH3-tails of one lipid - default is POPC.	108.6
ScatteringLengthOfOneMSP	cm	Scattering length of one belt protein - default is MSP1D1.	8.8E-10

Name	Unit	Description	Default
ScatteringLengthOfHeadgroup	cm	Scattering length of one lipid headgroup - default is POPC.	7.05E-12
ScatteringLengthOfCH2Tail	cm	Scattering length of the CH2-chains of one lipid - default is POPC.	-1.76E-12
ScatteringLengthOfCH3Tail	cm	Scattering length of the CH3-tails of one lipid - default is POPC.	-9.15E-13
Roughness		Factor used to smear the interfaces of the nanodisc.	5.0
Concentration	mM	Concentration of sample.	0.01
RhoSolvent	cm/Å^3	Scattering length density of solvent - default is D2O.	6.4e-14
AbsorptionCrosssection	1/m	Absorption cross section of the sample.	0.0
xwidth	m	Dimension of component in the x-direction.	
yheight	m	Dimension of component in the y-direction.	
zdepth	m	Dimension of component in the z-direction.	
SampleToDetectorDistance	m	Distance from sample to detector (for focusing the scattered neutrons).	
DetectorRadius	m	Radius of the detector (for focusing the scattered neutrons).	
qMin	Å^{-1}	Lowest q-value, for which a point is generated in the scattering profile	0.001
qMax	Å^{-1}	Highest q-value, for which a point is generated in the scattering profile	1.0

Links

- Component source code found in file `SANSNanodiscsFast.comp`.

14.83. The `SANSNanodiscsWithTags` McStas Component

A sample of monodisperse phospholipid bilayer nanodiscs in solution (water) - with histidine tag still on the belt proteins.

Identification

- **Author:** Martin Cramer Pedersen (mcpe@nbi.dk)
- **Origin:** KU-Science
- **Date:** October 17, 2012

Description

A component simulating the scattering from a box-shaped, thin solution (water) of monodisperse phospholipid bilayer nanodiscs - with histidine tags still on the belt proteins.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
AxisRatio		Axis ratio of the bilayer patch.	1.4
NumberOfLipids		Number of lipids per nanodisc.	130.0
AreaPerLipidHeadgroup	Å^2	Area per lipid headgroup - default is POPC.	65.0
HeightOfMSP	Å	Height of the belt protein - default is MSP1D1.	24.0
RadiusOfGyrationForHisTag	Å	Radius of gyration for the his-tag.	10.0
VolumeOfOneMSP	Å^3	Volume of one belt protein - default is MSP1D1.	26296.5
VolumeOfHeadgroup	Å^3	Volume of one lipid headgroup - default is POPC.	319.0
VolumeOfCH2Tail	Å^3	Volume of the CH2-chains of one lipid - default is POPC.	818.8
VolumeOfCH3Tail	Å^3	Volume of the CH3-tails of one lipid - default is POPC.	108.6
VolumeOfOneHisTag	Å^3	Volume of one his-tag.	2987.3
ScatteringLengthOfOneMSP	cm	Scattering length of one belt protein - default is MSP1D1.	8.8E-10
ScatteringLengthOfHeadgroup	cm	Scattering length of one lipid headgroup - default is POPC.	7.05E-12
ScatteringLengthOfCH2Tail	cm	Scattering length of the CH2-chains of one lipid - default is POPC.	-1.76E-12
ScatteringLengthOfCH3Tail	cm	Scattering length of the CH3-tails of one lipid - default is POPC.	-9.15E-13
ScatteringLengthOfOneHisTag	cm	Scattering length of one his-tag.	1.10E-10
Roughness		Factor used to smear the interfaces of the nanodisc.	5.0
Concentration	mM	Concentration of sample.	0.01
RhoSolvent	cm/Å^3	Scattering length density of solvent - default is D2O.	6.4e-14
AbsorptionCrosssection	1/m	Absorption cross section of the sample.	0.0
xwidth	m	Dimension of component in the x-direction.	

Name	Unit	Description	Default
yheight	m	Dimension of component in the y-direction.	
zdepth	m	Dimension of component in the z-direction.	
SampleToDetectorDistance	m	Distance from sample to detector (for focusing the scattered neutrons).	
DetectorRadius	m	Radius of the detector (for focusing the scattered neutrons).	

Links

- Component source code found in file `SANSNanodiscsWithTags.comp`.

14.84. The `SANSNanodiscsWithTagsFast` McStas Component

A sample of monodisperse phospholipid bilayer nanodiscs in solution (water) - with histidine tag still on the belt proteins.

Identification

- **Author:** Martin Cramer Pedersen (mcpe@nbi.dk)
- **Origin:** KU-Science
- **Date:** October 17, 2012

Description

A component very similar to EmptyNanodiscsWithTags.comp - however, the scattering profile is only computed once, and linear interpolation is the used to simulate the instrument.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
NumberOfQBins		Number of points generated in inital scattering profile.	200
AxisRatio		Axis ratio of the bilayer patch.	1.4
NumberOfLipids		Number of lipids per nanodisc.	130.0
AreaPerLipidHeadgroup	Å^2	Area per lipid headgroup - default is POPC.	65.0

Name	Unit	Description	Default
HeightOfMSP	Å	Height of the belt protein - default is MSP1D1.	24.0
RadiusOfGyrationForHisTag	Å	Radius of gyration for the his-tag.	12.7
VolumeOfOneMSP	$Å^3$	Volume of one belt protein - default is MSP1D1.	26296.5
VolumeOfHeadgroup	$Å^3$	Volume of one lipid headgroup - default is POPC.	319.0
VolumeOfCH2Tail	$Å^3$	Volume of the CH2-chains of one lipid - default is POPC.	818.8
VolumeOfCH3Tail	$Å^3$	Volume of the CH3-tails of one lipid - default is POPC.	108.6
VolumeOfOneHisTag	$Å^3$	Volume of one his-tag.	2987.3
ScatteringLengthOfOneMSP	cm	Scattering length of one belt protein - default is MSP1D1.	8.8E-10
ScatteringLengthOfHeadgroup	cm	Scattering length of one lipid headgroup - default is POPC.	7.05E-12
ScatteringLengthOfCH2Tail	cm	Scattering length of the CH2-chains of one lipid - default is POPC.	-1.76E-12
ScatteringLengthOfCH3Tail	cm	Scattering length of the CH3-tails of one lipid - default is POPC.	-9.15E-13
ScatteringLengthOfOneHisTag	cm	Scattering length of one his-tag.	1.10E-10
Roughness		Factor used to smear the interfaces of the nanodisc.	5.0
Concentration	mM	Concentration of sample.	0.01
RhoSolvent	$cm/Å^3$	Scattering length density of solvent - default is D2O.	6.4e-14
AbsorptionCrosssection	1/m	Absorption cross section of the sample.	0.0
xwidth	m	Dimension of component in the x-direction.	
yheight	m	Dimension of component in the y-direction.	
zdepth	m	Dimension of component in the z-direction.	
SampleToDetectorDistance	m	Distance from sample to detector (for focusing the scattered neutrons).	
DetectorRadius	m	Radius of the detector (for focusing the scattered neutrons).	
qMin	$Å^{-1}$	Lowest q-value, for which a point is generated in the scattering profile	0.001
qMax	$Å^{-1}$	Highest q-value, for which a point is generated in the scattering profile	1.0

Links

- Component source code found in file `SANSNanodiscsWithTagsFast.comp`.

14.85. The `SANSPDB` McStas Component

A sample describing a thin solution of proteins. This components must be compiled with the -lgsl and -lgslcblas flags (and possibly linked to the appropriate libraries).

Identification

- **Author:** Martin Cramer Pedersen (mcpe@nbi.dk) and Søren Kynde (kynde@nbi.dk)
- **Origin:** KU-Science
- **Date:** October 17, 2012

Description

This components expands the formfactor amplitude of the protein on spherical harmonics and computes the scattering profile using these. The expansion is done on amino-acid level and does not take hydration layer into account. The component must have a valid .pdb-file as an argument.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
RhoSolvent	Å	Scattering length density of the buffer.	9.4e-14
Concentration	mM	Concentration of sample.	0.01
AbsorptionCrosssection	1/m	Absorption cross section of the sample.	0.0
xwidth	m	Dimension of component in the x-direction.	
yheight	m	Dimension of component in the y-direction.	
zdepth	m	Dimension of component in the z-direction.	
SampleToDetectorDistance	m	Distance from sample to detector (for focusing the scattered neutrons).	
DetectorRadius	m	Radius of the detector (for focusing the scattered neutrons).	
PDBFilepath		Path to the file describing the high resolution structure of the protein.	"PDBfile.pdb"

Links

- Component source code found in file `SANSPDB.comp`.

14.86. The `SANSPDBFast` McStas Component

A sample describing a thin solution of proteins using linear interpolation to increase computational speed. This components must be compiled with the -lgsl and -lgslcblas flags (and possibly linked to the appropriate libraries).

Identification

- **Author:** Martin Cramer Pedersen (mcpe@nbi.dk) and Søren Kynde (kynde@nbi.dk)
- **Origin:** KU-Science
- **Date:** October 17, 2012

Description

This components expands the formfactor amplitude of the protein on spherical harmonics and computes the scattering profile using these. The expansion is done on amino-acid level and does not take hydration layer into account. The component must have a valid .pdb-file as an argument.

This is fast implementation of the SANSPDB sample component.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
RhoSolvent	Å	Scattering length density of the buffer - default is 100% D2O.	9.4e-14
Concentration	mM	Concentration of sample.	0.01
AbsorptionCrosssection	1/m	Absorption cross section of the sample.	0.0
xwidth	m	Dimension of component in the x-direction.	
yheight	m	Dimension of component in the y-direction.	
zdepth	m	Dimension of component in the z-direction.	
SampleToDetectorDistance		Distance from sample to detector (for focusing the scattered neutrons).	
DetectorRadius	m	Radius of the detector (for focusing the scattered neutrons).	

Name	Unit	Description	Default
qMin	$Å^{-1}$	Lowest q-value, for which a point is generated in the scattering profile	0.001
qMax	$Å^{-1}$	Highest q-value, for which a point is generated in the scattering profile	0.5
NumberOfQBins		Number of points generated in inital scattering profile.	200
PDBFilepath		Path to the file describing the high resolution structure of the protein.	"PDBfile.pdb"

Links

- Component source code found in file `SANSPDBFast.comp`.

14.87. The `SANSShells` McStas Component

A sample of monodisperse shell-like particles in solution.

Identification

- **Author:** Martin Cramer Pedersen (mcpe@nbi.dk)
- **Origin:** KU-Science
- **Date:** October 17, 2012

Description

A simple component simulating the scattering from a box-shaped, thin solution of monodisperse, shell-like particles.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
R	Å	Average radius of the particles.	100.0
Thickness	Å	Thickness of the shell - so that the outer radius is R + Thickness and the inner is R - Thickness.	5.0
Concentration	mM	Concentration of sample.	0.01
DeltaRho	$cm/Å^3$	Excess scattering length density of the particles.	1.0e-14
AbsorptionCrosssection	$1/m$	Absorption cross section of the sample.	0.0

Name	Unit	Description	Default
xwidth	m	Dimension of component in the x-direction.	
yheight	m	Dimension of component in the y-direction.	
zdepth	m	Dimension of component in the z-direction.	
SampleToDetectorDistance	m	Distance from sample to detector (for focusing the scattered neutrons).	
DetectorRadius	m	Radius of the detector (for focusing the scattered neutrons).	

Links

- Component source code found in file `SANSShells.comp`.

14.88. The `SANSSpheres` McStas Component

A sample of mono- or polydisperse spherical particles in solution.

Identification

- **Author:** Martin Cramer Pedersen (mcpe@nbi.dk)
- **Origin:** KU-Science
- **Date:** October 17, 2012

Description

A simple component simulating the scattering from a box-shaped, thin solution of monodisperse, spherical particles.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
R	Å	Radius of the spherical particles.	100.0
dR	Å	Variance of the radius of spherical particles. Default 0 (monodisperse spheres)	0.0
Concentration	mM	Concentration of sample.	0.01
DeltaRho	cm/Å^3	Excess scattering length density of the particles.	1.0e-14

Name	Unit	Description	Default
AbsorptionCrosssection	1/m	Absorption cross section of the sample.	0.0
xwidth	m	Dimension of component in the x-direction.	
yheight	m	Dimension of component in the y-direction.	
zdepth	m	Dimension of component in the z-direction.	
SampleToDetectorDistance		Distance from sample to detector (for focusing the scattered neutrons).	
DetectorRadius	m	Radius of the detector (for focusing the scattered neutrons).	

Links

- Component source code found in file `SANSSpheres.comp`.

14.89. The `SANSSpheresPolydisperse` McStas Component

A sample of mono- or polydisperse spherical particles in solution.

Identification

- **Author:** Linda Udby (udby@nbi.dk) - modified from SANSSpheres by Martin Cramer Pedersen (mcpe@nbi.dk)
- **Origin:** KU-Science
- **Date:** October 17, 2012

Description

A simple component simulating the scattering from a box-shaped, thin solution of monodisperse, spherical particles.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
R	Å	Radius of the spherical particles.	100.0
dR	Å	Variance of the radius of spherical particles. Default 0 (monodisperse spheres)	0.0
Concentration	mM	Concentration of sample.	0.01

Name	Unit	Description	Default
DeltaRho	cm/Å^3	Excess scattering length density of the particles.	1.0e-14
AbsorptionCrosssection	1/m	Absorption cross section of the sample.	0.0
xwidth	m	Dimension of component in the x-direction.	
yheight	m	Dimension of component in the y-direction.	
zdepth	m	Dimension of component in the z-direction.	
SampleToDetectorDistance	m	Distance from sample to detector (for focusing the scattered neutrons).	
DetectorRadius	m	Radius of the detector (for focusing the scattered neutrons).	

Links

- Component source code found in file `SANSSpheresPolydisperse.comp`.

14.90. The `SANS_AnySamp` McStas Component

Sample for Small Angle Neutron Scattering. To be customized.

Identification

- **Author:** Henrich Frielinghaus
- **Origin:** FZ-Juelich/FRJ-2/IFF/KWS-2
- **Date:** Sept 2004

Description

Sample for Small Angle Neutron Scattering. May be customized for Any Sample model. Copy this component and modify it to your needs. Just give shape of scattering function. Normalization of scattering will be done in INITIALIZE.

Some remarks: Scattering function should be a defined the same way in INITIALIZE and TRACE sections There exist maybe better library functions for the integral.

Here for comparison: Guinier. Transmitted paths set to 3% of all paths. In this simulation method paths are well distributed among transmission and scattering (equally in Q-space).

Sample components leave the units of flux for the probability of the individual paths. That is more consitent than the Sans_spheres routine. Furthermore one can simulate the transmitted beam. This allows to determine the needed size of the beam stop. Only

absorption has not been included yet to these sample-components. But thats really nothing.

Example: SANS_AnySamp(transm=0.5, Rg=100, qmax=0.03, xwidth=0.01, yheight=0.01, zdepth=0.001)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
transm	1	coherent transmission of sample for the optical path "zdepth"	0.5
Rg	Å	Radius of Gyration	100
qmax	$Å^{-1}$	Maximum scattering vector	0.03
xwidth	m	horizontal dimension of sample, as a width	0.01
yheight	m	vertical dimension of sample, as a height	0.01
zdepth	m	thickness of sample	0.001

Links

- Component source code found in file `SANS_AnySamp.comp`.
- Sans_spheres component

14.91. The `SANS_DebyeS` McStas Component

Sample for Small Angle Neutron Scattering, Debye-Scherrer Ring

Identification

- **Author:** Henrich Frielinghaus
- **Origin:** FZ-Juelich/FRJ-2/IFF/KWS-2
- **Date:** Sept 2004

Description

Debye-Scherrer Ring: Model for highly ordered diblock copolymer. This example is highly simple. Multiple scattering neglected, but could be incorporated. Sample that only produces a DebyeSherrer ring. Advantages: Transmitted beam is simulated. Intensities still in flux units. Minor point: Absolute scattering probability given by transmission.

Sample components leave the units of flux for the probability of the individual paths. That is more consitent than the Sans_spheres routine. Furthermore one can simulate

the transmitted beam. This allows to determine the needed size of the beam stop. Only absorption has not been included yet to these sample-components. But thats really nothing.

Example: SANS_DebyeS(transm=0.5, qDS=0.008, xwidth=0.01, yheight=0.01, zdepth=0.001)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
transm	1	coherent transmission of sample for the optical path "zdepth"	0.5
qDS	$Å^{-1}$	Scattering vector of Debye-Sherrer ring	0.008
xwidth	m	horiz. dimension of sample, as a width (m)	0.01
yheight	m	vert.. dimension of sample, as a height (m)	0.01
zdepth	m	thickness of sample (m)	0.001

Links

- Component source code found in file `SANS_DebyeS.comp`.
- Sans_spheres component

14.92. The `SANS_Guinier` McStas Component

Sample for Small Angle Neutron Scattering, Guinier model

Identification

- **Author:** Henrich Frielinghaus
- **Origin:** FZ-Juelich/FRJ-2/IFF/KWS-2
- **Date:** Sept 2004

Description

Sample that scatters with a Guinier shape. This is just an example where analytically an integral exists. The neutron paths are proportional to the intensity (low intensity > few paths).

Guinier function (Rg) a = Rg*Rg/3 propability_unscaled = q * exp(-a*q*q) integral_prop_unscal = 1/(2*a) * (1 - exp(-a*q*q))

```
propability_scaled   = 2*a * q*exp(-a*q*q) / (1 - exp(-a*q*q))
```

integral_prop_scaled = (1 - exp(-a*q*q)) / (1 - exp(-a*qmax*qmax))

In this simulation method many paths occur for high propability. For simulation of low intensities see SANS_AnySamp.

Sample components leave the units of flux for the probability of the individual paths. That is more consitent than the Sans_spheres routine. Furthermore one can simulate the transmitted beam. This allows to determine the needed size of the beam stop. Only absorption has not been included yet to these sample-components. But thats really nothing.

Example: SANS_Guinier(transm=0.5, Rg=100, qmax=0.03, xwidth=0.01, yheight=0.01, zdepth=0.001)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
transm	1	(coherent) transmission of sample for the optical path "zdepth"	0.5
Rg	Å	Radius of Gyration	100
qmax	$Å^{-1}$	Maximum scattering vector	0.03
xwidth	m	horiz. dimension of sample, as a width	0.01
yheight	m	vert.. dimension of sample, as a height	0.01
zdepth	m	thickness of sample	0.001

Links

- Component source code found in file `SANS_Guinier.comp`.
- Sans_spheres component

14.93. The `SANS_Liposomes_Abs` McStas Component

Solid dilute monodisperse spheres sample with transmitted beam for Small Angle Neutron Scattering.

Identification

- **Author:** Wim Bouwman, Delft University of Technology
- **Origin:** TUDelft
- **Date:** Aug 2012

Description

Sample for Small Angle Neutron Scattering. Modified from the Any Sample model. Normalization of scattering is done in INITIALIZE. Some elements of Sans_spheres have been re-used

Some remarks: Scattering function should be a defined the same way in INITIALIZE and TRACE sections There exist maybe better library functions for the integral.

Transmitted paths set to 50% of all paths to be optimised for SESANS. In this simulation method paths are well distributed among transmission and scattering (equally in Q-space).

Sample components leave the units of flux for the probability of the individual paths. That is more consitent than the Sans_spheres routine. Furthermore one can simulate the transmitted beam. This allows to determine the needed size of the beam stop. Absorption is included to these sample-components.

Example: SANS_Spheres_Abs(R=100, Phi=1e-3, Delta_rho=0.6, sigma_a = 50, qmax=0.03, xwidth=0.01, yheight=0.01, zthick=0.001)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
R	Å	Radius of scattering hard spheres	100
dR	Å	Gaussian standard deviation of the radius R (polydispersity)	0.0
Phi	1	Particle volume fraction	1e-3
Delta_rho	fm/Å^3	Excess scattering length density	0.6
sigma_a	m^{-1}	Absorption cross section density at 2200 m/s	0.50
qmax	Å^{-1}	Maximum scattering vector (typically 10/R to observe the 3rd ring)	0.03
xwidth	m	horiz. dimension of sample, as a width	0.01
yheight	m	vert.. dimension of sample, as a height	0.01
zthick	m	thickness of sample	0.001
dbilayer	Å	Thickness of the lipid bilayer (shell)	35

Links

- Component source code found in file `SANS_Liposomes_Abs.comp`.
- Sans_spheres component
- SANS_AnySamp component

14.94. The SANS_benchmark2 McStas Component

SANS_benchmark2 sample from FZ Jülichx

Identification

- **Author:** Henrich Frielinghaus
- **Origin:** FZJ
- **Date:** Apr 2013

Description

Several benchmark SANS samples are defined in this routine. The first ones are analytically defined. Higher numbers are forseen for tables. In principle, the exact definitions can be changed freely - inside this code. The consideration of all parameters as a routine parameter would be too much for the general purpose. The user might decide to make single parameters routine parameters.

For the scattering simulation a high fraction of neutron paths is directed to the scattering (exact fraction is sc_aim). The remaining paths are used for the transmitted beams. The absolute intensities are treated accordingly, and the p-parameter is set accordingly.

For the scattering probability, the integral of the scattering function between Q = 0.0001 and 1.0 Å-1 is calculated. This is used in terms of transmisson, and of course for the scattering probability. In this way, multiple scattering processes could be treated as well.

The typical SANS range was considered to be between 0.0001 and 1.0 Å-1. This means that the scattered neutrons are equally distributed in this range on logarithmic Q-scales. The Q-parameters can be changed still inside the code, if needed.

Example: SANS_benchmark(xwidth=0.01, yheight=0.01, zthick=0.001, model=1.0, dsdw_inc=0.02, sc_aim=0.97, sans_aim=0.95, singlesp = 0.0)

The component includes 19 sample-model-settings: case 0 - no coherent scattering case 1 - polymer with Mw = 2.000g/mol case 2 - polymer with Mw = 1.000.000g/mol case 3 - microemulsion case 4 - wormlike micelle case 5 - sphere, R = 25 Å case 6 - sphere, R = 500 Å case 7 - polymer blend case 8 - diblock copolymer case 9 - multilamellar vesicles case 10 - logarithmically spaced series of sharp peaks case 11 - logarithmically spaced series of sharp delta peaks !!!!!! ... case 15 - sphere, R = 150 Å

```
case 18   free
```

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xwidth	m	width of sample volume	0.01
yheight	m	height of sample volume	0.01
zthick	m	thickness of sample volume	0.001
model	1	model no., real variable will be rounded. negative numbers interpreted as model #0, too large as #18. See above list for details.	1.0
dsdw_inc	1	the incoherent background from the overall sample (cm-1), should read ca. 1.0 for water, 0.5 for half D2O, half H2O, and ca. 0.02 for D2O	0.02
sc_aim	1	the fraction of neutron paths used to represent the scattered neutrons (including everything: incoherent and coherent). rest is transmission.	0.97
sans_aim	1	the fraction of neutron paths used to represent the scattered neutrons in the sans-range (up to 1.0AA-1). rest is incoherent with Q>1AA-1.	0.95
singlesp	1	switches between multiple scattering (parameter zero 0.0) and single scattering (parameter 1.0). The sc_aim directs a fraction of paths to the first scattering process accordingly. The no. of paths for the second scattering process is derived from the real probability. Up to 10 scattering processes are considered.	0.0

Links

- Component source code found in file `SANS_benchmark2.comp`.

14.95. The `Sans_liposomes_new` McStas Component

Release: McStas 1.12

SANS sample, spherical shells in thin solution with gaussian size distribution

Identification

- **Author:** P. Willendrup, K. Lefmann, L. Arleth, L. Udby
- **Origin:** Risoe

- **Date:** 19.12.2003

Description

Sample for Small Angle Neutron Scattering - spherical shells in thin solution with gaussian size distribution The shape of the sample may be a cylinder of given radius or a box with dimensions xwidth, yheight, zthick. qmax defines a cutoff in q.

Example: Sans_spheres(R = 100, dR = 5, dbilayer=35, Phi = 1e-3, Delta_rho = 0.6, sigma_a = 50, qmax = 0.3 ,xwidth=0.01, yheight=0.01, zthick=0.005)

%BUGS This component does NOT simulate absolute intensities. This latter depends on the detector parameters.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
R	Å	Radius of scattering hard spheres	100
dR	Å	Gaussian standard deviation of the radius R (polydispersity)	0.0
Phi	1	Particle volume fraction	1e-3
Delta_rho	fm/Å^3	Excess scattering length density	0.6
sigma_a	m^{-1}	Absorption cross section density at 2200 m/s	0.50
dist	m	Distance from sample to detector	6
Rdet	m	Detector (disk-shaped) radius	0.5
xwidth	m	horiz. dimension of sample, as a width	0.01
yheight	m	vert . dimension of sample, as a height	0.01
zthick	m	thickness of sample	0.005
qmax	Å^{-1}	Maximum momentum transfer	0
dbilayer	Å	Thickness of shell	35

Links

- Component source code found in file `Sans_liposomes_new.comp`.
- The test/example instrument SANS.instr.

14.96. The `Multilayer_Sample` McStas Component

Multilayer Reflecting sample using matrix Formula, v2 with 'nrepeats' for repeating multilayer structure.

Identification

- **Author:** Robert Dalgliesh
- **Origin:** ISIS
- **Date:** June 2010

Description

in order to get this to compile you need to link against the gsl and gslcblas libraries.
to do this automatically edit /usr/local/lib/mcstas/tools/perl/mcstas_config.perl
add -lgsl and -lgslcblas to the CFLAGS line
Horizontal reflecting substrate defined by SLDs,Thicknesses, roughnesses The superphase may also be determined
Example: Multilayer_Sample(xmin=-0.1, xmax=0.1,zmin=-0.1, zmax=0.1, nlayer=1,sldPar={0.0,2.0e-6,0.0e-6},dPar={20.0}, sigmaPar={5.0,5.0})
Example: d1 500: sld1 (air) 0.0: sld2 (Si) 2.07e-6: sldf1(film Ni) 9.1e-6

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xwidth	m	Width of substrate	0.2
zlength	m	Length of substrate	0.2
nlayer	1	Number of film layers	1
nrepeats	1	Number of repeats of the block of layers appart from the superphase and substrate	1
sldPar	$Å^{-2}$	Scattering length Density's of layers	{0.0,2.0e-6,0.0e-6}
dPar	Å	Thicknesses of film layers	{20.0}
sigmaPar	Å	r.m.s roughnesses of the interfaces	{5.0,5.0}
frac_inc	1	Fraction of statistics to assign to incoherent scattering	0
ythick	m	Thickness of substrate	0
mu_inc	cm^{-1}	Macroscopic incoherent scattering cross section of the substrate	5.62
mu_abs	cm^{-1}	Macroscopic absorption cross section of the substrate at 2200 m/s (scales as 1/v). Together with mu_inc it attenuates the incoherently scattered neutrons	0

Name	Unit	Description	Default
target_index	1	Used in combination with focus_xw and focus_yh to indicate solid angle for incoherent scattering.	0
focus_xw	m	Used in combination with target_index and focus_yh to indicate solid angle for incoherent scattering.	0

Links

- Component source code found in file `Multilayer_Sample.comp`.

14.97. Contributed monitors and detectors

14.98. The `E_4PI` McStas Component

Spherical Energy-sensitive detector

Identification

- **Author:** Duc Le
- **Origin:** ISIS
- **Date:** 2000s

Description

Example: E_4PI(filename="e4pi.dat",ne=200,Emin=0,Emax=E*2)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
ne	1	Number of energy bins	50
Emin	meV	Minimum energy measured	0
Emax	meV	Maximum energy measured	5
filename	string	Name of file in which to store the output data	0
radius	m	Radius of detector (m)	1
restore_neutron	1	If set, the monitor does not influence the neutron state	0

Links

- Component source code found in file `E_4PI.comp`.
- <A HREF="http://neutron.risoe.dk/mcstas/components/tests/powder/">Test
- results</A> (not up-to-date).

14.99. The `NPI_tof_dhkl_detector` McStas Component

Release: McStas 2.5 Last update: 30/11/2018
NPI detector for estimating dhkl from tof

Identification

- **Author:** Jan Saroun, saroun@ujf.cas.cz
- **Origin:** NPI
- **Date:** July 1, 2017

Description

A cylindrical detector which converts time-of-flight data (x,y,z,time) to a 1D diffractogram in dhkl. The detector model includes finite detection depth, spatial and time resolution. Optionally, the component exports position and time coordinates of detection events in an ASCII file.

The component can also handle setups employing a modulation chopper for peak multiplexing at a long pulse source, as proposed for the diffractometer BEER@ESS. In this case, the component requires chopper parameters (modulation period, width of the primary unmodulated pulse), and a file with estimated dhkl values. The component then estimates valid regions on the angle/tof map, excluding areas assumed to be empty or with overlaping lines. This map is exported together with the diffractogram.

Tips for usage: 1) Centre the detector at the sample axis, keep z-axis parallel to the incident beam. 2) Set the radius equal to the distance from the sample axis to the front face of the detection volume. 3) Linst-Lc is used to determine "instrumental" wavelength as lambda = h/m_n*time/(Linst-Lc);

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename	str	Name of file in which to store the detector data.	0
radius	m	Radius of detector.	2

Name	Unit	Description	Default
yheight	m	Height of detector.	0.3
zdepth	m	Thickness of the detection volume.	0.01
amin	deg	minimum of scattering angle to be detected.	75
amax	deg	maximum of scattering angle to be detected.	105
nd		Number of bins for dhkl in the 1D diffractograms.	200
na		Number of bins for scattering angles in the 2D maps.	800
nt		Number of bins for time of flight in the 2D maps.	800
nev		Number of detection events to export in "events.dat" (only modulation mode, set 1 for no export)	1
d_min	Å	minimum of inter-planar spacing to be detected.	1
d_max	Å	maximum of inter-planar spacing to be detected.	3
time0	s	Time delay of the wavelength definition chopper with respect to the source pulse.	
Linst	m	Distance from the source to the detector.	
Lc	m	Distance from the source to the pulse chopper (set 0 for a short pulse source).	
res_x	m	Spatial resolution along x (Gaussian FWHM).	0
res_y	m	Spatial resolution along y (Gaussian FWHM).	0
res_t	s	Time resolution (Gaussian FWHM). Readout only, path to capture is accounted for by the tracing code.	0
mu	1/cm/Å	Capture coefficient for the detection medium.	1.0
mod_shift	float	Relative shift of the modulation pattern (delta_d/d, constant for all reflections).	0
modulation	1|0	Switch on/off the modulation regime (BEER multiplexing technique).	0
mod_dt	s	Modulation period.	0
mod_twidth	s	Width of the primary source pulse.	0
mod_d0_table	str	Name of a file with the list of dhkl estimates (one per line).	0
verbose	1|0	Switch for extended reporting.	0

Name	Unit	Description	Default
restore_neutron	1|0	Switch for restoring of previous neutron state.	1

Links

- Component source code found in file `NPI_tof_dhkl_detector.comp`.

14.100. The `NPI_tof_theta_monitor` McStas Component

Release: McStas 1.6
Cylindrical (2pi) PSD Time-of-flight monitor.

Identification

- **Author:** Kim Lefmann
- **Origin:** Risoe
- **Date:** October 2000

Description

Derived from TOF_cylPSD_monitor. Code is extended by the option allowing to define range of scattering angles, therefore creating only a part of the cylinder surface. The plot is transposed when compared to TOF_cylPSD_monitor: scattering angles are on the horizontal axis.

Example: TOF_cylPSD_monitor_NPI(nt = 1024, nphi = 960, filename = "Output.dat", radius = 2, yheight = 1.0, tmin = 3e4, tmax = 12e4, amin = 75, amax = 105, restore_neutron = 1)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename	str	Name of file in which to store the detector image	0
radius	m	Cylinder radius	1
yheight	m	Cylinder height	0.3
tmin	mu-s	Beginning of time window	
tmax	mu-s	End of time window	
amin	deg	minimum angle to detect	
amax	deg	maximum angle to detect	

Name	Unit	Description	Default
restore_neutron	1	If set, the monitor does not influence the neutron state	1
verbose	1	If non-zero, print additional information	0
nt	1	Number of time bins	128
na	1	Number of angular bins	90

Links

- Component source code found in file `NPI_tof_theta_monitor.comp`.

14.101. The `PSD_Detector` McStas Component

Position-sensitive gas-filled detector with gaseous thermal-neutron converter (box, cylinder or 'banana').

Identification

- **Author:** Thorwald van Vuure
- **Origin:** ILL
- **Date:** Feb 21st, 2005

Description

An n times m pixel Position Sensitive Detector (PSD), box, cylinder or banana filled with a mixture of thermal-neutron converter gas and stopping gas. This model implements wires detection, including charge drift, parallax, etc. The default is a simple 1D/2D position detector. However, setting the parameter 'type' to "events" will save the signal as an event file which contains all neutron parameters at detection points, including time. This is especially suited for TOF detectors. Please use Histogrammer afterwards. The gas creation event (neutron absorption in gas) is flagged as SCATTERED==1, and the charge detection (signal) is flagged as SCATTERED==2.

GEOMETRY: A flat rectangular box of given depth is simulated, or alternatively a cylinder (when giving radius), or a horizontal cylindrical 'banana' detector (when giving the width along the arc). Box position is at its center, the 'banana' and cylinder have their position at focus point, symetric in angular range for the 'banana'.

GAS SPECIFICATIONS: The converter gas can be specified as He-3, BF3 or N2. The filling pressure of converter gas must be specified along with the zdepth: this gives an absorption probability (e.g. ~90% for 0.18 nm neutrons at zdepth 3 cm and He-3 pressure 5 bar). The stopping gas can be specified as one of the following gases: CF4, C3H8, CO2, Xe/TMA or He, or any for which a table of stopping powers is available. NB: each stopping gas table has specific data on a pair of particles (from thermal-neutron

absorptions in either He-3, BF3 or N2) in a specific gas. Unusual choices like N2 as a converter and He as a stopping gas probably require the calculation of a new table. Tip: to do a simulation in pure He-3, specify 0 bar for the stopping gas. The pressures of the two gases together put a lower limit on the achievable spatial resolution, no matter what pixel size is specified. Example: 1 bar CF4 gives ~2.5 mm spatial resolution, 3 bar CF4 gives ~1 mm. C3H8 requires more pressure for the same resolution, then Xe/TMA, then CO2 and finally He.

IMPLEMENTED FEATURES: Taken into account are the following effects: - (rectangularly distributed) error in the recorded position due to the range of the neutron capture products in the gas - angular dependence of absorption efficiency - border effect (events outside the sensitive volume tend to be counted in the border pixels) - parallax effect (with, if desired, an electrostatic lens to partially correct for this - use "LensOn=1") - wall effect To correctly take the wall effect into account, a value must be specified for the energy threshold for acceptance of an event as a valid neutron event. This normally serves to discriminate from gammas. Because these are not simulated in McStas, a value of 0 keV can be chosen for this threshold if one is interested in seeing the maximum number of neutrons; however, a more realistic value would be 100 keV. This implies seeing the maximum number of neutrons as well, except in geometries with dominating wall effect. The border effect is a considerably complex phenomenon: it depends on the precise electric field configuration near the border. It is simulated here simply by introducing a dead zone, gas-filled, bordering the sensitive volume. Events in there can still cause signals in the detector. The dead zone can cause a shadow in the sensitive volume, just as in reality. The algorithm simply adds all events on the first 'pixel' in the dead zone to the border pixels. This is a coarse approximation, because the physical effect depends on the orientation of the tracks and the energy threshold setting. The bottom line is that tracking the position of the Center Of Gravity of the charge deposition in the detector does not allow for accurate modeling of the border effect, and therefore the border pixels should be ignored, just as in a real detector. Note that specifying 0 width of the dead zone next to the sensitive volume, apart from being unrealistic, does not allow accurate simulation of the border effect: in this case, there will be a relative lack of events on the border pixels due to the wall effect. This component determines the position of the Center Of Gravity of the charge cloud in the detector. It does not simulate independent channels. Manufacturing imperfections such as wire diameter variations and spread in electronic amplifier component properties are not simulated. This should allow an experimenter to identify these manufacturing imperfections in his physical machine and correct for them.

Example: PSD_Detector(xwidth=0.192, yheight=0.192, nx=64, ny=64, zdepth=0.03, threshold=100, borderx=-1, bordery=-1, PressureConv=5, PressureStop=1, FN_Conv="Gas_tables/He3inHe.ta FN_Stop="Gas_tables/He3inCF4.table", xChDivRelSigma=0, yChDivRelSigma=0, filename="BIDIM19.psd")

Example: PSD_Detector(xwidth=0.26, yheight=0.26, nx=128, ny=128, zdepth=0.03, threshold=100, borderx=-1, bordery=-1, PressureConv=5, PressureStop=1, FN_Conv="Gas_tables/He3inHe.ta FN_Stop="Gas_tables/He3inCF4.table", xChDivRelSigma=0, yChDivRelSigma=0, filename="BIDIM26.psd")

Example: PSD_Detector(radius=0.0127, yheight=0.20, nx=1, ny=128, threshold=100, borderx=-1, PressureConv=9.9, PressureStop=0.1, FN_Conv="Gas_tables/He3inHe.table", FN_Stop="Gas_tables/He3inCO2.table", yChDivRelSigma=0.006, filename="ReuterStokes.psd")
Example: PSD_Detector(awidth=1.533, yheight=0.4, nx=640, ny=256, radius=0.73, zdepth=0.032, dc=0.026, threshold=100, borderx=-1, bordery=-1, PressureConv=5, PressureStop=1, FN_Conv="Gas_tables/He3inHe.table", FN_Stop="Gas_tables/He3inCF4.table", xChDivRelSigma=0, yChDivRelSigma=0.0037, LensOn=1, filename="D19.psd")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nx	1	Number of pixel columns	128
ny	1	Number of pixel rows	128
xwidth	m	Width of detector opening, in the case of a flat box	0
radius	m	Radius of detector for cylindrical/banana shape	0
awidth	m	Width of detector opening along the horizontal arc of the detector, measured along the inside arc of the sensitive volume in the case of a 'banana' detector	0
yheight	m	Height of detector opening	0
zdepth	m	Depth of the sensitive volume of the detector	0.03
threshold	keV	Energy threshold for acceptance of an event	100
PressureConv	bar	Pressure of gaseous thermal-neutron converter (e.g. He-3)	5
PressureStop	bar	Pressure of stopping gas	1
interpolate	1	Performs energy interpolation if true	1
p_interact	1	Fraction of statistical events to be treated by component. Set it to 0 to use real absorption probability	0
verbose	1	Gives more information and spectrum	0
LensOn	1	set to 1 to simulate an electrostatic lens according to P. Van Esch, NIM A 540 (2005) pp. 361-367.	1
dc	m	Distance from the entrance window to the cathode plane, where the correction zone of the lens ends for banana shape and LensOn=1	0

Name	Unit	Description	Default
borderx	m	Width of (gas-filled) border on the side of the detector	-2
bordery	m	Height of (gas-filled) border on the top/bottom of the detector giving rise to various border effects. Set to -2 to obtain the height of one pixel, -1 for the depth of the detector (default), or specify a non-negativedistance	-2
xChDivRelSigma	1	Sigma of the error in position determination along the x axis (relative to xwidth), uniquely due to charge division readout. Set to 0 in case of independent channel readout along the x dimension	0
yChDivRelSigma	1	Relative sigma due to charge division in y direction, relative to yheight.	0
bufsize	1	Size of neutron output buffer default is 0, i.e. save all - recommended.	0
restore_neutron	1	If set, the monitor does not influence the neutron state	0
angle	deg	Angular opening of detector in the case of a 'banana' detector, then overrides awidth parameter	0
type	string	If set to 'events' detector signal saves signal into an event file to be read e.g. by Histogrammer. This is to be used for TOF detectors.	0
filename	text	Name of file in which to store the detector image. The name of the component is used if file name is not specified	0
FN_Conv	text	Filename of the stopping power table of the converter gas	"He3inHe.table"
FN_Stop	text	Filename of the stopping power table of the stopping gas	"He3inCF4.table"

Links

- Component source code found in file `PSD_Detector.comp`.
- The test/example instrument Test_PSD_Detector.instr.

14.102. The `PSD_monitor_Filter` McStas Component

Position-sensitive monitor with filter-option, used in Radiography_Sword instrument.

Identification

- **Author:** Viktor L. Holm
- **Origin:** University of Copenhagen
- **Date:** 2019

Description

An (n times m) pixel PSD monitor with filter-option, adapted from PSD_monitor from Kim Lefmann.

Example: PSD_monitor_Filter(xmin=-0.1, xmax=0.1, ymin=-0.1, ymax=0.1, nx=90, ny=90, filename="Output.psd")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nx	1	Number of pixel columns	90
ny	1	Number of pixel rows	90
filter_pixels	1	Number of filter pixels (in both x and y)	600
filename	str	Filename for storing the detector image	0
xmin	m	Lower x bound of detector opening	-0.05
xmax	m	Upper x bound of detector opening	0.05
ymin	m	Lower y bound of detector opening	-0.05
ymax	m	Upper y bound of detector opening	0.05
xwidth	m	Width of detector. Overrides xmin, xmax	0
yheight	m	Height of detector. Overrides ymin, ymax	0
restore_neutron	1	If set, the monitor does not influence the neutron state	0
nowritefile	1	If set, monitor will skip writing to disk	0
TwoD_filename	str	Filter inputfile	0
apply_xy_filter	1	Flag to apply filter or not.	1
filter_xmin	m	Lower x bound of filter dimensions	-0.8
filter_xmax	m	Upper x bound of filter dimensions	0.8
filter_ymin	m	Lower y bound of filter dimensions	-0.8
filter_ymax	m	Upper y bound of filter dimensions	0.8

Name	Unit	Description	Default
pinhole_filter_distanc	m	Distance between upstream pinhole and the filter	25
pinhole_detector_distance	m	Distance between upstream pinhole and the detector	25
smoothing	1	Level of smoothing to apply (0-1)*rand_norm()*(pinhole_filter_distance / pinhole_detector_distance)	0.0
zoom	1	Level of detector-zoom	1.0

Links

- Component source code found in file `PSD_monitor_Filter.comp`.

14.103. The `PSD_monitor_rad` McStas Component

Position-sensitive monitor with radially averaging.

Identification

- **Author:** Henrich Frielinghaus
- **Origin:** FZ-Juelich/FRJ-2/IFF/KWS-2
- **Date:** Sept 2004

Description

Radial monitor that allows for radial averaging. Comment: The intensity is given as two files: 1) a radial sum 2) a radial average (i.e. intensity per area)
Example: PSD_monitor_rad(rmax=0.2, nr=100, filename="Output.psd", filename_av="Output_av.psd")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nr	1	Number of concentric circles	100
filename	text	Name of file in which to store the detector image	0
filename_av	text	Name of file in which to store the averaged detector image	0
rmax	m	Outer radius of detector	0.2

Links

- Component source code found in file `PSD_monitor_rad.comp`.

14.104. The `Radial_div` McStas Component

A radial divergence sensitive monitor with wavelength restrictions.

Identification

- **Author:** Kaspar Hewitt Klenø, modified from Hdiv_monitor
- **Origin:** NBI
- **Date:** Jan 2011

Description

A radial divergence sensitive monitor with wavelength restrictions. The counts are distributed in n pixels.

Example:

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
ndiv	1	Number of pixel rows	20
restore_neutron	1	If set, the monitor does not influence the neutron state	0
filename	str	Name of file in which to store the detector image	
xmin	m	Lower x bound of detector opening	0
xmax	m	Upper x bound of detector opening	0
ymin	m	Lower y bound of detector opening	0
ymax	m	Upper y bound of detector opening	0
xwidth	m	Width/diameter of detector (x). Overrides xmin,xmax.	0
yheight	m	Height of detector (y). Overrides ymin,ymax.	0
maxdiv_r	deg	Maximal radial divergence detected	2
Lmin	Å	Minimum wavelength detected	
Lmax	Å	Maximum wavelength detected	

Links

- Component source code found in file `Radial_div.comp`.

14.105. The `SANSQMonitor` McStas Component

A circular detector measuring the radial average of intensity as a function of the momentum transform in the sample.

Identification

- **Author:** Søren Kynde (kynde@nbi.dk) and Martin Cramer Pedersen (mcpe@nbi.dk)
- **Origin:** KU-Science
- **Date:** October 17, 2012

Description

A simple component simulating the scattering from a box-shaped, thin solution of monodisperse, spherical particles.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
NumberOfBins		Number of bins in the r (and q).	100
RFilename		File used for storing I(r).	"RDetector"
qFilename		File used for storing I(q).	"QDetector"
RadiusDetector	m	Radius of the detector (in the xy-plane).	
DistanceFromSample	m	Distance from the sample to this component.	
LambdaMin	Å	Max sensitivity in lambda - used to compute the highest possible value of momentum transfer, q.	1.0
Lambda0	Å	If lambda is given, the momentum transfers of all rays are computed from this value. Otherwise, instrumental effects are neglected.	0.0
restore_neutron		If set to 1, the component restores the original neutron.	0

Links

- Component source code found in file `SANSQMonitor.comp`.

14.106. The `TOFSANSdet` McStas Component

Multiple TOF detectors for SANS instrument. The component is to be placed at the sample position. For the time being better switch gravity off.

Identification

- **Author:** Henrich Frielinghaus, FZJuelich
- **Origin:** FZJ
- **Date:** Apr 2013

Description

TOF monitor that calculates I of q.
Example: TOFSANSdet();

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
Nq	1	Number of q-bins	100
plength	s	Pulse length	0.00286
ssdist	m	Source-Sample distance for TOF calculation.	27.0
coldis	m	Collimation length	20.0
Sthckn	cm	sample thickness	0.1
ds1	m	distance of detector 1	1.0
xw1	m	width of detector 1 (0 for off)	0.0
yh1	m	height of detector 1 (0 for off)	0.0
hl1	m	w/h of hole in det 1 (0 for off), full width	0.0
ds2	m	distance.....2	5.0
xw2	m	width.....2	1.0
yh2	m	heigth.....2	1.0
hl2	m	hole.....2	0.2
ds3	m	distance of detector 3	20.0
xw3	m	width.....3	1.0
yh3	m	height.....3	1.0
hl3	m	hole.....3 (beam stop, used for primary beam detection)	0.04
vx3		vertical extension of beam stop down (in case of gravity off set 0.0, otherwise value larger than 1.0)	0.0

Name	Unit	Description	Default
tmin	s	Beginning of time window	0.005
tmax	s	End of time window	0.15
Nx		Number of horizontal detector pixels (detector 1,2,3) better leave unchanged	128.0
Ny		Number of vertical detector pixels (detector 1,2,3) better leave unchanged	128.0
Nt		Number of time bins (detector 1,2,3) better leave unchanged	500.0
qmin	1/A	Lower limit of q-range	0.0005
qmax	1/A	Upper limit of q-range	0.11
rstneu		restore neutron after treatment ??? (0.0 = no)	0.0
centol		tolerance of center determination (if center larger than centol*calculated_center then set back to theory)	0.1
inttol		tolerance of intensity (if primary beam intensity smaller than inttol*max_intensity then discard data)	0.0001
qcal		calibration of intensity (cps) to width of q-bin (non_zero = yes, zero = no)	1
fname		file name (first part without extensions)	0

Links

- Component source code found in file `TOFSANSdet.comp`.

14.107. The `TOF2Q_cyl_monitor` McStas Component

Release: McStas 2.0

Cylindrical (2pi) 2theta v. q Time-of-flight monitor.

Identification

- **Author:** Kim Lefmann
- **Origin:** Risoe
- **Date:** October 2000

Description

Cylindrical (2pi) 2theta v. q Time-of-flight monitor.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
nq	1	Number of q bins	100
nphi	1	Number of angular bins	100
nh	1	Number of bins in detector height	10
restore_neutron	1	If set, the monitor does not influence the neutron state	0
filename	str	Name of file in which to store the detector image	0
radius	m	Cylinder radius	1.0
height	m	Cylinder height	0.1
q_min	1/Å	Minimum q value	0
q_max	1/Å	Maximum q value	20
L_chop2samp	m	Distance from resolution chopper to sample position	144.0
t_chop	ms	Flight time of mean wave length from source to resolution chopper	0.004
pixel	1	Flag, 0: no pixelation and perfect time. 1: make pixels from nphi and nh and time resolution limitation.	1

Links

- Component source code found in file `TOF2Q_cyl_monitor.comp`.

14.108. The `TOF_PSDmonitor` McStas Component

Position-sensitive TOF vs. (height) linear monitor.

Identification

- **Author:** Robert Dalgliesh
- **Origin:** ISIS
- **Date:** September 2021

Description

Adapted from Kim Lefmann's TOF_cylPSDmonitor and PSD_monitors An n pixel (vertical, linear) PSD monitor with Time of Flight detection. This component may also be used as a beam detector.

Example: TOF_PSDmonitor(xmin=-0.1, xmax=0.1, ymin=-0.1, ymax=0.1, ny=90, nchan=100, TOFmin=0.0, TOFmax=20000.0, filename="Output.psd")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
ny	1	Number of pixel rows	90
nchan	1	Number of TOF channels	100
xwidth	m	width of detector opening	0.1
xmin	m	Lower x bound of detector opening	0
xmax	m	Upper x bound of detector opening	0
yheight	m	height of detector opening	0.1
ymin	m	Lower y bound of detector opening	0
ymax	m	Upper y bound of detector opening	0
TOFmin	mu-s	Beginning TOF window	0
TOFmax	mu-s	End TO window	1e6
nowritefile	1	If set, detector will skip writing to disk	0
filename	str	Name of file in which to store the detector image	0

Links

- Component source code found in file `TOF_PSDmonitor.comp`.

14.109. The `TOF_PSDmonitor_toQ` McStas Component

Position-sensitive, linear Q monitor.

Identification

- **Author:** Robert Dalgliesh
- **Origin:** ISIS
- **Date:** September 2021

Description

One-dimensional Q-monitor. Calculates Q from "measured" time-of-flight, i.e. not from particle velocity parameters.

Adapted from Kim Lefmann's TOF_cylPSDmonitor and PSD_monitors An n pixel Q-monitor based on measured Time of Flight detection. This component may also be used as a beam detector.

Example: TOF_PSDmonitor_toQ(xmin=-0.1, xmax=0.1, ymin=-0.1, ymax=0.1, ny=90, nqbin=100, TOFmin=0.0, TOFmax=20000.0, filename="Output.psd")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
ny	1	Number of y bins (for discretisation of vertical detector dimension into "bins")	90
nqbin	1	number of q bins	500
xwidth	m	width of detector opening	0.1
xmin	m	Lower x bound of detector opening	0
xmax	m	Upper x bound of detector opening	0
yheight	m	height of detector opening	0.1
ymin	m	Lower y bound of detector opening	0
ymax	m	Upper y bound of detector opening	0
tmin	mu-s	Beginning TOF window	0.0
tmax	mu-s	End TO window	100000.0
qmin	$Å^{-1}$	Start of q window	0.005
qmax	$Å^{-1}$	End of q window	0.5
L1	m	distance from source to sample (m)	23.0
L2	m	distance from sample to detector (m)	3.0
detTheta	deg	Nominal detector theta	0.91
filename	str	Name of file in which to store the detector image	0
nowritefile	1	If set, detector will skip writing to disk	0

Links

- Component source code found in file `TOF_PSDmonitor_toQ.comp`.

14.110. Other contributed components

14.111. The `Wavelength_Diffuser` McStas Component

Wavelength-diffuser: Applies Gaussian smearing in wavelength.

Identification

- **Author:** Viktor L. Holm
- **Origin:** University of Copenhagen
- **Date:** Aug 13, 2019

Description

A component that applies a Gaussian smearing in wavelength to the incoming neutron. The smearing is defined by a normal-distributed uncertainty in wavelength from 0 to +/- dlambda.

Example: Wavelength_Diffuser(dlambda = 1)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
dlambda	Å	Sigma-value for Gaussian wavelength-smearing	1

Links

- Component source code found in file `Wavelength_Diffuser.comp`.

14.112. Contributed MCViNE sample components

The components in this section are McStas ports, contributed by Fahima Islam, of sample-scattering kernels from the MCViNE neutron ray-tracing package [Lin+15][1]. They cover scatterers defined by an analytic expression or a tabulated grid for the structure factor $S(Q)$ or the dynamical structure factor $S(Q, E)$, isotropic and single-crystal dispersions with optional Gaussian or Lorentzian broadening, scatterers with fixed energy and/or momentum transfer, one-phonon scattering from a tabulated phonon dispersion, a resolution kernel for direct-geometry spectrometers, and a 2D SANS map on a grid. Every kernel exists in two forms sharing the same kernel code: as an ordinary sample component (this section), and as a Union process with the suffix `_process` (next section).

14.113. The `MCViNE_SQ` McStas Component

Elastic isotropic scatterer with structure factor S(|Q|) (expression or table).

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE SQkernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/SQkernel.cc (+ GridSQ, SQ_fromexpression))
- **Date:** 2026-09-29

[1] `https://github.com/mcvine/mcvine`; preprint at `https://arxiv.org/abs/1504.02776`

Description

Elastic scattering with angular distribution given by S(|Q|), for liquids, glasses or powder-averaged diffuse scattering. S(Q) is either an expression in Q or a 2-column text file (Q [Å^-1], S), linearly interpolated (0 outside).

Functions are given as strings evaluated at run time (MCViNE uses fparser): + - * / ^ (or **), % , comparisons, && ||, if(c,a,b), sin cos tan asin acos atan sinh cosh tanh exp log log10 log2 sqrt abs floor ceil int sign cbrt, pow atan2 min max hypot fmod, constants pi and e.

Transport (same as MCViNE HomogeneousNeutronScatterer): on the first passage the neutron is forced to scatter at a uniformly chosen depth, weighted by path length, attenuation exp(-(mu+sigma)x) and the scattering coefficient; on exit it is attenuated by exp(-(mu+sigma)L_out). With order>1 further scatterings are sampled analogically (truncated exponential). With p_transmit>0 that fraction of events is kept as the attenuated direct beam. mu(v) = absorption_coefficient*2200/v. Events for which the kernel cannot scatter (kinematically forbidden) are absorbed.

Geometry: box (xwidth,yheight,zdepth), cylinder (radius,yheight), hollow cylinder (+thickness) or sphere (radius only). All vectors (Q, targets, reciprocal vectors, atom positions) are in the component's local frame.

Uses the shared runtime share/mcvine-lib.h/.c.

Example: MCViNE_SQ(SQ="1+sin(Q*3)/(Q*3)", Qmin=0, Qmax=20, scattering_coefficient=10 radius=0.01, yheight=0.05)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
SQ	str	S(Q) expression (ignored if SQ_file given)	"1"
SQ_file	str	2-column file Q S	0
Qmin	$Å^{-1}$	Lower Q bound	0
Qmax	$Å^{-1}$	Upper Q bound	100
absorption_coefficient	m^{-1}	Absorption coefficient at 2200 m/s (MCViNE convention; scales as 1/v)	0
scattering_coefficient	m^{-1}	Scattering coefficient	0
sigma_abs	barn	Alternative: absorption cross section per unit cell at 2200 m/s (used when Vc>0)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0, coefficients are computed from sigma_abs, sigma_scat	0

Name	Unit	Description	Default
radius	m	Outer radius of a cylinder (with yheight) or of a sphere (yheight=0)	0
xwidth	m	Width of a box sample	0
yheight	m	Height of a box or cylinder sample	0
zdepth	m	Depth of a box sample	0
thickness	m	Wall thickness of a hollow cylinder (0: filled)	0
pack	1	Packing factor (scales absorption and scattering coefficients)	1
p_transmit	1	Monte Carlo fraction of events kept as unscattered transmitted beam (0: always scatter)	0
order	1	Maximum number of scattering events per neutron (1: single scattering)	1

Links

- Component source code found in file `MCViNE_SQ.comp`.
- MCViNE documentation: https://mcvine.github.io

14.114. The `MCViNE_SvQ` McStas Component

Elastic scatterer with a vector structure factor S(Qx,Qy,Qz) (expression or 3D grid).

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE SvQkernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/SvQkernel.cc (+ GridSvQ))
- **Date:** 2026-09-29

Description

Elastic scattering into a uniformly random direction, weighted by S(Q) with Q = ki - kf in the sample frame. S is either an expression in Qx,Qy,Qz or a 3D grid file: three rows 'min max n' for Qx, Qy, Qz followed by nx*ny*nz values (Qx slowest, Qz fastest), trilinearly interpolated (0 outside).

Functions are given as strings evaluated at run time (MCViNE uses fparser): + - * / ^ (or **), % , comparisons, && ||, if(c,a,b), sin cos tan asin acos atan sinh cosh tanh exp

log log10 log2 sqrt abs floor ceil int sign cbrt, pow atan2 min max hypot fmod, constants pi and e.

Transport (same as MCViNE HomogeneousNeutronScatterer): on the first passage the neutron is forced to scatter at a uniformly chosen depth, weighted by path length, attenuation exp(-(mu+sigma)x) and the scattering coefficient; on exit it is attenuated by exp(-(mu+sigma)L_out). With order>1 further scatterings are sampled analogically (truncated exponential). With p_transmit>0 that fraction of events is kept as the attenuated direct beam. mu(v) = absorption_coefficient*2200/v. Events for which the kernel cannot scatter (kinematically forbidden) are absorbed.

Geometry: box (xwidth,yheight,zdepth), cylinder (radius,yheight), hollow cylinder (+thickness) or sphere (radius only). All vectors (Q, targets, reciprocal vectors, atom positions) are in the component's local frame.

Uses the shared runtime share/mcvine-lib.h/.c.

Example: MCViNE_SvQ(SvQ="exp(-((Qx-1)^2+Qy^2+Qz^2)/0.01)", scattering_coefficient=10 xwidth=0.01, yheight=0.01, zdepth=0.01)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
SvQ	str	S(Qx,Qy,Qz) expression (ignored if SvQ_file given)	"1"
SvQ_file	str	3D grid file (see description)	0
absorption_coefficient	m^{-1}	Absorption coefficient at 2200 m/s (MCViNE convention; scales as 1/v)	0
scattering_coefficient	m^{-1}	Scattering coefficient	0
sigma_abs	barn	Alternative: absorption cross section per unit cell at 2200 m/s (used when Vc>0)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0, coefficients are computed from sigma_abs, sigma_scat	0
radius	m	Outer radius of a cylinder (with yheight) or of a sphere (yheight=0)	0
xwidth	m	Width of a box sample	0
yheight	m	Height of a box or cylinder sample	0
zdepth	m	Depth of a box sample	0
thickness	m	Wall thickness of a hollow cylinder (0: filled)	0
pack	1	Packing factor (scales absorption and scattering coefficients)	1

Name	Unit	Description	Default
p_transmit	1	Monte Carlo fraction of events kept as unscattered transmitted beam (0: always scatter)	0
order	1	Maximum number of scattering events per neutron (1: single scattering)	1

Links

- Component source code found in file `MCViNE_SvQ.comp`.
- MCViNE documentation: https://mcvine.github.io

14.115. The `MCViNE_SQE` McStas Component

Isotropic S(Q,E) scatterer from an analytic expression or grid, with optional final-energy focusing.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE SQEkernel / SQE_EnergyFocusing_Kernel (+ SQE_fromexpression); MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/SQEkernel.cc, SQE_EnergyFocusing_Kernel.cc)
- **Date:** 2026-09-29

Description

Isotropic (powder / liquid) S(|Q|,E). E is sampled uniformly in [Emin, min(Ei,Emax)] and |Q| uniformly in the kinematically allowed part of [Qmin,Qmax]. S(Q,E) is an expression in Q [Å^-1] and E [meV], or a grid file in the Isotropic_Sqw format (first numeric row: q values; second: energy values; then nq rows of nw values), bilinearly interpolated. For grid data McStas' Isotropic_Sqw already exists; the new parts are the analytic S(Q,E) and the energy focusing: when dEf>0, E is restricted so that Ef lies in [Ef-dEf, Ef+dEf] (MCViNE SQE_EnergyFocusing_Kernel).

Functions are given as strings evaluated at run time (MCViNE uses fparser): + - * / ^ (or **), % , comparisons, && ||, if(c,a,b), sin cos tan asin acos atan sinh cosh tanh exp log log10 log2 sqrt abs floor ceil int sign cbrt, pow atan2 min max hypot fmod, constants pi and e.

Transport (same as MCViNE HomogeneousNeutronScatterer): on the first passage the neutron is forced to scatter at a uniformly chosen depth, weighted by path length, attenuation exp(-(mu+sigma)x) and the scattering coefficient; on exit it is attenuated

by exp(-(mu+sigma)L_out). With order>1 further scatterings are sampled analogically (truncated exponential). With p_transmit>0 that fraction of events is kept as the attenuated direct beam. mu(v) = absorption_coefficient*2200/v. Events for which the kernel cannot scatter (kinematically forbidden) are absorbed.

Geometry: box (xwidth,yheight,zdepth), cylinder (radius,yheight), hollow cylinder (+thickness) or sphere (radius only). All vectors (Q, targets, reciprocal vectors, atom positions) are in the component's local frame.

Uses the shared runtime share/mcvine-lib.h/.c.

Example: MCViNE_SQE(SQE="exp(-(E-10)^2/2)*Q^2*exp(-Q^2/4)", Qmin=0, Qmax=10, Emin=-5, Emax=30, sigma_scat=5, sigma_abs=0.5, Vc=50, radius=0.01, yheight=0.05)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
SQE	str	S(Q,E) expression (ignored if SQE_file given)	"1"
SQE_file	str	S(q,w) grid file (Isotropic_Sqw layout)	0
Qmin	$Å^{-1}$	Lower Q bound	0
Qmax	$Å^{-1}$	Upper Q bound	10
Emin	meV	Lower energy transfer bound	-10
Emax	meV	Upper energy transfer bound	10
Ef	meV	Final-energy focusing centre (used when dEf>0)	0
dEf	meV	Final-energy focusing half range; 0 disables focusing	0
absorption_coefficient	m^{-1}	Absorption coefficient at 2200 m/s (MCViNE convention; scales as 1/v)	0
scattering_coefficient	m^{-1}	Scattering coefficient	0
sigma_abs	barn	Alternative: absorption cross section per unit cell at 2200 m/s (used when Vc>0)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0, coefficients are computed from sigma_abs, sigma_scat	0
radius	m	Outer radius of a cylinder (with yheight) or of a sphere (yheight=0)	0
xwidth	m	Width of a box sample	0
yheight	m	Height of a box or cylinder sample	0
zdepth	m	Depth of a box sample	0

Name	Unit	Description	Default
thickness	m	Wall thickness of a hollow cylinder (0: filled)	0
pack	1	Packing factor (scales absorption and scattering coefficients)	1
p_transmit	1	Monte Carlo fraction of events kept as unscattered transmitted beam (0: always scatter)	0
order	1	Maximum number of scattering events per neutron (1: single scattering)	1

Links

- Component source code found in file `MCViNE_SQE.comp`.
- MCViNE documentation: https://mcvine.github.io

14.116. The `MCViNE_E_Q` McStas Component

Isotropic dispersion: S(Q,E) = S(Q) delta(E - E(Q)) with analytic E(Q), S(Q).

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE E_Q_Kernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/E_Q_Kernel.ic
- **Date:** 2026-09-29

Description

Powder/liquid-like excitation with dispersion E(|Q|) given as an expression in Q [Å^-1] returning meV, and intensity S(Q). |Q| is sampled uniformly in [Qmin,Qmax] (up to 100 attempts to find a kinematically allowed Q).

Functions are given as strings evaluated at run time (MCViNE uses fparser): + - * / ^ (or **), % , comparisons, && ||, if(c,a,b), sin cos tan asin acos atan sinh cosh tanh exp log log10 log2 sqrt abs floor ceil int sign cbrt, pow atan2 min max hypot fmod, constants pi and e.

Transport (same as MCViNE HomogeneousNeutronScatterer): on the first passage the neutron is forced to scatter at a uniformly chosen depth, weighted by path length, attenuation exp(-(mu+sigma)x) and the scattering coefficient; on exit it is attenuated by exp(-(mu+sigma)L_out). With order>1 further scatterings are sampled analogically

(truncated exponential). With p_transmit>0 that fraction of events is kept as the attenuated direct beam. mu(v) = absorption_coefficient*2200/v. Events for which the kernel cannot scatter (kinematically forbidden) are absorbed.

Geometry: box (xwidth,yheight,zdepth), cylinder (radius,yheight), hollow cylinder (+thickness) or sphere (radius only). All vectors (Q, targets, reciprocal vectors, atom positions) are in the component's local frame.

Uses the shared runtime share/mcvine-lib.h/.c.

Example: MCViNE_E_Q(E_Q="20*sin(Q*1.5)^2", S_Q="1", Qmin=0, Qmax=10, scattering_coefficient=10, radius=0.01, yheight=0.05)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
E_Q	str	E(Q) expression, variable Q [Å^-1], result [meV]	"10"
S_Q	str	S(Q) expression, variable Q	"1"
Qmin	$Å^{-1}$	Lower Q bound	0
Qmax	$Å^{-1}$	Upper Q bound	10
unbiased	1	0: MCViNE sampling (up to 100 retries, weight/attempts; slightly biased when part of [Qmin,Qmax] is forbidden). 1: single attempt, unbiased	0
absorption_coefficient	m^{-1}	Absorption coefficient at 2200 m/s (MCViNE convention; scales as 1/v)	0
scattering_coefficient	m^{-1}	Scattering coefficient	0
sigma_abs	barn	Alternative: absorption cross section per unit cell at 2200 m/s (used when Vc>0)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0, coefficients are computed from sigma_abs, sigma_scat	0
radius	m	Outer radius of a cylinder (with yheight) or of a sphere (yheight=0)	0
xwidth	m	Width of a box sample	0
yheight	m	Height of a box or cylinder sample	0
zdepth	m	Depth of a box sample	0
thickness	m	Wall thickness of a hollow cylinder (0: filled)	0
pack	1	Packing factor (scales absorption and scattering coefficients)	1

Name	Unit	Description	Default
p_transmit	1	Monte Carlo fraction of events kept as unscattered transmitted beam (0: always scatter)	0
order	1	Maximum number of scattering events per neutron (1: single scattering)	1

Links

- Component source code found in file `MCViNE_E_Q.comp`.
- MCViNE documentation: https://mcvine.github.io

14.117. The `MCViNE_E_vQ` McStas Component

Single-crystal dispersion: S(Q,E) = S(Q) delta(E - E(Q)) with analytic E(Qx,Qy,Qz).

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE E_vQ_Kernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/E_vQ_Kernel.
- **Date:** 2026-09-29

Description

Excitation with dispersion E(Qx,Qy,Qz) [meV] and intensity S(Qx,Qy,Qz), given as expressions of the Q vector in the sample (component) frame [Å^-1]. For a random final direction, all solutions kf of Ei - Ef(kf) = E(ki - kf) with Ef in [Ei-Emax, Ei] are found (nsteps sub-intervals + Ridders' method) and one is picked; the weight contains the Jacobian of the delta function.

Functions are given as strings evaluated at run time (MCViNE uses fparser): + - * / ^ (or **), % , comparisons, && ||, if(c,a,b), sin cos tan asin acos atan sinh cosh tanh exp log log10 log2 sqrt abs floor ceil int sign cbrt, pow atan2 min max hypot fmod, constants pi and e.

Transport (same as MCViNE HomogeneousNeutronScatterer): on the first passage the neutron is forced to scatter at a uniformly chosen depth, weighted by path length, attenuation exp(-(mu+sigma)x) and the scattering coefficient; on exit it is attenuated by exp(-(mu+sigma)L_out). With order>1 further scatterings are sampled analogically (truncated exponential). With p_transmit>0 that fraction of events is kept as the attenuated direct beam. mu(v) = absorption_coefficient*2200/v. Events for which the kernel cannot scatter (kinematically forbidden) are absorbed.

Geometry: box (xwidth,yheight,zdepth), cylinder (radius,yheight), hollow cylinder (+thickness) or sphere (radius only). All vectors (Q, targets, reciprocal vectors, atom positions) are in the component's local frame.

Uses the shared runtime share/mcvine-lib.h/.c.

Example: MCViNE_E_vQ(E_Q="20*(sin(Qx*1.57)^2+sin(Qy*1.57)^2+sin(Qz*1.57)^2)", S_Q="1", Emax=60, scattering_coefficient=10, xwidth=0.02, yheight=0.02, zdepth=0.02)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
E_Q	str	E(Qx,Qy,Qz) expression [meV]	"10"
S_Q	str	S(Qx,Qy,Qz) expression	"1"
Emax	meV	Maximum energy transfer considered	10
nsteps	1	Number of sub-intervals for the kf root search (MCViNE: 1000)	1000
xacc	$Å^{-1}$	Root accuracy in kf (MCViNE: 1e-3)	1e-3
absorption_coefficient	m^{-1}	Absorption coefficient at 2200 m/s (MCViNE convention; scales as 1/v)	0
scattering_coefficient	m^{-1}	Scattering coefficient	0
sigma_abs	barn	Alternative: absorption cross section per unit cell at 2200 m/s (used when Vc>0)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0, coefficients are computed from sigma_abs, sigma_scat	0
radius	m	Outer radius of a cylinder (with yheight) or of a sphere (yheight=0)	0
xwidth	m	Width of a box sample	0
yheight	m	Height of a box or cylinder sample	0
zdepth	m	Depth of a box sample	0
thickness	m	Wall thickness of a hollow cylinder (0: filled)	0
pack	1	Packing factor (scales absorption and scattering coefficients)	1
p_transmit	1	Monte Carlo fraction of events kept as unscattered transmitted beam (0: always scatter)	0
order	1	Maximum number of scattering events per neutron (1: single scattering)	1

Links

- Component source code found in file `MCViNE_E_vQ.comp`.
- MCViNE documentation: https://mcvine.github.io

14.118. The `MCViNE_Broadened_E_Q` McStas Component

Isotropic dispersion E(Q) broadened by a Gaussian of Q-dependent width.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE Broadened_E_Q_Kernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/Broadened_E_
- **Date:** 2026-09-29

Description

S(Q,E) = S(Q) G(E - E(Q); sigma(Q)) with G a normalised Gaussian of standard deviation sigma(Q). Q is sampled uniformly in [Qmin,Qmax]; the energy offset is sampled from the line shape (up to 100 attempts). Neutrons with Ei below min(E(Q)-3w(Q)) do not scatter.

Functions are given as strings evaluated at run time (MCViNE uses fparser): + - * / ^ (or **), % , comparisons, && ||, if(c,a,b), sin cos tan asin acos atan sinh cosh tanh exp log log10 log2 sqrt abs floor ceil int sign cbrt, pow atan2 min max hypot fmod, constants pi and e.

Transport (same as MCViNE HomogeneousNeutronScatterer): on the first passage the neutron is forced to scatter at a uniformly chosen depth, weighted by path length, attenuation exp(-(mu+sigma)x) and the scattering coefficient; on exit it is attenuated by exp(-(mu+sigma)L_out). With order>1 further scatterings are sampled analogically (truncated exponential). With p_transmit>0 that fraction of events is kept as the attenuated direct beam. mu(v) = absorption_coefficient*2200/v. Events for which the kernel cannot scatter (kinematically forbidden) are absorbed.

Geometry: box (xwidth,yheight,zdepth), cylinder (radius,yheight), hollow cylinder (+thickness) or sphere (radius only). All vectors (Q, targets, reciprocal vectors, atom positions) are in the component's local frame.

Uses the shared runtime share/mcvine-lib.h/.c.

Example: MCViNE_Broadened_E_Q(E_Q="20*sin(Q*1.5)^2", S_Q="1", sigma_Q="0.5", Qmin=0, Qmax=10, scattering_coefficient=10, radius=0.01, yheight=0.05)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
E_Q	str	E(Q) expression [meV], variable Q [Å^-1]	"10"
S_Q	str	S(Q) expression	"1"
sigma_Q	str	Gaussian sigma(Q) [meV]	"1"
Qmin	$Å^{-1}$	Lower Q bound	0
Qmax	$Å^{-1}$	Upper Q bound	10
unbiased	1	0: MCViNE sampling (retries without acceptance correction: over-weights when part of the Q range is forbidden). 1: single attempt, unbiased	0
absorption_coefficient	m^{-1}	Absorption coefficient at 2200 m/s (MCViNE convention; scales as 1/v)	0
scattering_coefficient	m^{-1}	Scattering coefficient	0
sigma_abs	barn	Alternative: absorption cross section per unit cell at 2200 m/s (used when Vc>0)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0, coefficients are computed from sigma_abs, sigma_scat	0
radius	m	Outer radius of a cylinder (with yheight) or of a sphere (yheight=0)	0
xwidth	m	Width of a box sample	0
yheight	m	Height of a box or cylinder sample	0
zdepth	m	Depth of a box sample	0
thickness	m	Wall thickness of a hollow cylinder (0: filled)	0
pack	1	Packing factor (scales absorption and scattering coefficients)	1
p_transmit	1	Monte Carlo fraction of events kept as unscattered transmitted beam (0: always scatter)	0
order	1	Maximum number of scattering events per neutron (1: single scattering)	1

Links

- Component source code found in file `MCViNE_Broadened_E_Q.comp`.
- MCViNE documentation: https://mcvine.github.io

14.119. The `MCViNE_LorentzianBroadened_E_Q` McStas Component

Isotropic dispersion E(Q) broadened by a Lorentzian of Q-dependent width.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE LorentzianBroadened_E_Q_Kernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/LorentzianBroa
- **Date:** 2026-09-29

Description

S(Q,E) = S(Q) L(E - E(Q); gamma(Q)) with L a Lorentzian of half width at half maximum gamma(Q). Q is sampled uniformly in [Qmin,Qmax]; the energy offset is sampled from the line shape (up to 100 attempts). Neutrons with Ei below min(E(Q)-3w(Q)) do not scatter.

Functions are given as strings evaluated at run time (MCViNE uses fparser): + - * / ^ (or **), % , comparisons, && ||, if(c,a,b), sin cos tan asin acos atan sinh cosh tanh exp log log10 log2 sqrt abs floor ceil int sign cbrt, pow atan2 min max hypot fmod, constants pi and e.

Transport (same as MCViNE HomogeneousNeutronScatterer): on the first passage the neutron is forced to scatter at a uniformly chosen depth, weighted by path length, attenuation exp(-(mu+sigma)x) and the scattering coefficient; on exit it is attenuated by exp(-(mu+sigma)L_out). With order>1 further scatterings are sampled analogically (truncated exponential). With p_transmit>0 that fraction of events is kept as the attenuated direct beam. mu(v) = absorption_coefficient*2200/v. Events for which the kernel cannot scatter (kinematically forbidden) are absorbed.

Geometry: box (xwidth,yheight,zdepth), cylinder (radius,yheight), hollow cylinder (+thickness) or sphere (radius only). All vectors (Q, targets, reciprocal vectors, atom positions) are in the component's local frame.

Uses the shared runtime share/mcvine-lib.h/.c.

Example: MCViNE_LorentzianBroadened_E_Q(E_Q="20*sin(Q*1.5)^2", S_Q="1", gamma_Q="0.5", Qmin=0, Qmax=10, scattering_coefficient=10, radius=0.01, yheight=0.05)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
E_Q	str	E(Q) expression [meV], variable Q [Å^-1]	"10"
S_Q	str	S(Q) expression	"1"

Name	Unit	Description	Default
gamma_Q	str	Lorentzian HWHM gamma(Q) [meV]	"1"
Qmin	$Å^{-1}$	Lower Q bound	0
Qmax	$Å^{-1}$	Upper Q bound	10
unbiased	1	0: MCViNE sampling (retries without acceptance correction: over-weights when part of the Q range is forbidden). 1: single attempt, unbiased	0
absorption_coefficient	m^{-1}	Absorption coefficient at 2200 m/s (MCViNE convention; scales as 1/v)	0
scattering_coefficient	m^{-1}	Scattering coefficient	0
sigma_abs	barn	Alternative: absorption cross section per unit cell at 2200 m/s (used when Vc>0)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0, coefficients are computed from sigma_abs, sigma_scat	0
radius	m	Outer radius of a cylinder (with yheight) or of a sphere (yheight=0)	0
xwidth	m	Width of a box sample	0
yheight	m	Height of a box or cylinder sample	0
zdepth	m	Depth of a box sample	0
thickness	m	Wall thickness of a hollow cylinder (0: filled)	0
pack	1	Packing factor (scales absorption and scattering coefficients)	1
p_transmit	1	Monte Carlo fraction of events kept as unscattered transmitted beam (0: always scatter)	0
order	1	Maximum number of scattering events per neutron (1: single scattering)	1

Links

- Component source code found in file `MCViNE_LorentzianBroadened_E_Q.comp`.
- MCViNE documentation: https://mcvine.github.io

14.120. The `MCViNE_ConstantEnergyTransfer` McStas Component

Isotropic scatterer with a fixed energy transfer.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE ConstantEnergyTransferKernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/ConstantEnerg
- **Date:** 2026-09-29

Description

Every scattered neutron loses exactly the energy E (E<0: gains) and leaves in a uniformly random direction over 4pi. Useful for testing and resolution studies.

Transport (same as MCViNE HomogeneousNeutronScatterer): on the first passage the neutron is forced to scatter at a uniformly chosen depth, weighted by path length, attenuation exp(-(mu+sigma)x) and the scattering coefficient; on exit it is attenuated by exp(-(mu+sigma)L_out). With order>1 further scatterings are sampled analogically (truncated exponential). With p_transmit>0 that fraction of events is kept as the attenuated direct beam. mu(v) = absorption_coefficient*2200/v. Events for which the kernel cannot scatter (kinematically forbidden) are absorbed.

Geometry: box (xwidth,yheight,zdepth), cylinder (radius,yheight), hollow cylinder (+thickness) or sphere (radius only). All vectors (Q, targets, reciprocal vectors, atom positions) are in the component's local frame.

Uses the shared runtime share/mcvine-lib.h/.c.

Example: MCViNE_ConstantEnergyTransfer(E=10, scattering_coefficient=10, radius=0.01, yheight=0.05)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
E	meV	Energy transfer Ei-Ef	0
absorption_coefficient	m^{-1}	Absorption coefficient at 2200 m/s (MCViNE convention; scales as 1/v)	0
scattering_coefficient	m^{-1}	Scattering coefficient	0
sigma_abs	barn	Alternative: absorption cross section per unit cell at 2200 m/s (used when Vc>0)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0, coefficients are computed from sigma_abs, sigma_scat	0
radius	m	Outer radius of a cylinder (with yheight) or of a sphere (yheight=0)	0

Name	Unit	Description	Default
xwidth	m	Width of a box sample	0
yheight	m	Height of a box or cylinder sample	0
zdepth	m	Depth of a box sample	0
thickness	m	Wall thickness of a hollow cylinder (0: filled)	0
pack	1	Packing factor (scales absorption and scattering coefficients)	1
p_transmit	1	Monte Carlo fraction of events kept as unscattered transmitted beam (0: always scatter)	0
order	1	Maximum number of scattering events per neutron (1: single scattering)	1

Links

- Component source code found in file `MCViNE_ConstantEnergyTransfer.comp`.
- MCViNE documentation: https://mcvine.github.io

14.121. The `MCViNE_ConstantQE` McStas Component

Scatterer with fixed momentum transfer |Q| and energy transfer E.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE ConstantQEKernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/Co
- **Date:** 2026-09-29

Description

Scattered neutrons have energy transfer E and |Q| fixed; the azimuth around ki is random. S(Q,E) = delta(|Q|-Q0) delta(E-E0) (unnormalised, as in MCViNE).

Transport (same as MCViNE HomogeneousNeutronScatterer): on the first passage the neutron is forced to scatter at a uniformly chosen depth, weighted by path length, attenuation exp(-(mu+sigma)x) and the scattering coefficient; on exit it is attenuated by exp(-(mu+sigma)L_out). With order>1 further scatterings are sampled analogically (truncated exponential). With p_transmit>0 that fraction of events is kept as the attenuated direct beam. mu(v) = absorption_coefficient*2200/v. Events for which the kernel cannot scatter (kinematically forbidden) are absorbed.

Geometry: box (xwidth,yheight,zdepth), cylinder (radius,yheight), hollow cylinder (+thickness) or sphere (radius only). All vectors (Q, targets, reciprocal vectors, atom positions) are in the component's local frame.

Uses the shared runtime share/mcvine-lib.h/.c.

Example: MCViNE_ConstantQE(Q=3, E=20, scattering_coefficient=10, xwidth=0.02, yheight=0.05, zdepth=0.002)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
Q	$Å^{-1}$	Momentum transfer |Q|	1
E	meV	Energy transfer	0
absorption_coefficient	m^{-1}	Absorption coefficient at 2200 m/s (MCViNE convention; scales as 1/v)	0
scattering_coefficient	m^{-1}	Scattering coefficient	0
sigma_abs	barn	Alternative: absorption cross section per unit cell at 2200 m/s (used when Vc>0)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0, coefficients are computed from sigma_abs, sigma_scat	0
radius	m	Outer radius of a cylinder (with yheight) or of a sphere (yheight=0)	0
xwidth	m	Width of a box sample	0
yheight	m	Height of a box or cylinder sample	0
zdepth	m	Depth of a box sample	0
thickness	m	Wall thickness of a hollow cylinder (0: filled)	0
pack	1	Packing factor (scales absorption and scattering coefficients)	1
p_transmit	1	Monte Carlo fraction of events kept as unscattered transmitted beam (0: always scatter)	0
order	1	Maximum number of scattering events per neutron (1: single scattering)	1

Links

- Component source code found in file `MCViNE_ConstantQE.comp`.
- MCViNE documentation: https://mcvine.github.io

14.122. The MCViNE_ConstantvQE McStas Component

Scatterer with a fixed momentum-transfer vector Q, weighted by a Gaussian in E.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE ConstantvQEKernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/Co
- **Date:** 2026-09-29

Description

kf = ki - Q (Q vector fixed in the sample frame); the event weight is multiplied by exp(-(E-E0)^2/(2 dE^2)) where E = Ei-Ef. Used for single-crystal resolution studies.

Transport (same as MCViNE HomogeneousNeutronScatterer): on the first passage the neutron is forced to scatter at a uniformly chosen depth, weighted by path length, attenuation exp(-(mu+sigma)x) and the scattering coefficient; on exit it is attenuated by exp(-(mu+sigma)L_out). With order>1 further scatterings are sampled analogically (truncated exponential). With p_transmit>0 that fraction of events is kept as the attenuated direct beam. mu(v) = absorption_coefficient*2200/v. Events for which the kernel cannot scatter (kinematically forbidden) are absorbed.

Geometry: box (xwidth,yheight,zdepth), cylinder (radius,yheight), hollow cylinder (+thickness) or sphere (radius only). All vectors (Q, targets, reciprocal vectors, atom positions) are in the component's local frame.

Uses the shared runtime share/mcvine-lib.h/.c.

Example: MCViNE_ConstantvQE(Qx=1, Qy=0, Qz=0.5, E=10, dE=1, scattering_coefficient=10 xwidth=0.01, yheight=0.01, zdepth=0.01)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
Qx	$Å^{-1}$	Q vector x	0
Qy	$Å^{-1}$	Q vector y	0
Qz	$Å^{-1}$	Q vector z	0
E	meV	Nominal energy transfer	0
dE	meV	Gaussian sigma of the energy weight	1
absorption_coefficient	m^{-1}	Absorption coefficient at 2200 m/s (MCViNE convention; scales as 1/v)	0
scattering_coefficient	m^{-1}	Scattering coefficient	0

Name	Unit	Description	Default
sigma_abs	barn	Alternative: absorption cross section per unit cell at 2200 m/s (used when Vc>0)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	Å^3	Unit cell volume; when >0, coefficients are computed from sigma_abs, sigma_scat	0
radius	m	Outer radius of a cylinder (with yheight) or of a sphere (yheight=0)	0
xwidth	m	Width of a box sample	0
yheight	m	Height of a box or cylinder sample	0
zdepth	m	Depth of a box sample	0
thickness	m	Wall thickness of a hollow cylinder (0: filled)	0
pack	1	Packing factor (scales absorption and scattering coefficients)	1
p_transmit	1	Monte Carlo fraction of events kept as unscattered transmitted beam (0: always scatter)	0
order	1	Maximum number of scattering events per neutron (1: single scattering)	1

Links

- Component source code found in file `MCViNE_ConstantvQE.comp`.
- MCViNE documentation: https://mcvine.github.io

14.123. The `MCViNE_Phonon_IncoherentElastic` McStas Component

Incoherent elastic scattering with Debye-Waller factor exp(-2W), 2W = dw_core*Q^2.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE phonon IncoherentElastic kernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/phonon/Incohe
- **Date:** 2026-09-29

Description

Isotropic incoherent elastic scattering damped by the Debye-Waller factor. dw_core [Å^2] is given directly, or (dw_core<0) computed from a phonon DOS, temperature and average atomic mass exactly as MCViNE's DWFromDOS. McStas' Incoherent has no Debye-Waller factor.

Transport (same as MCViNE HomogeneousNeutronScatterer): on the first passage the neutron is forced to scatter at a uniformly chosen depth, weighted by path length, attenuation exp(-(mu+sigma)x) and the scattering coefficient; on exit it is attenuated by exp(-(mu+sigma)L_out). With order>1 further scatterings are sampled analogically (truncated exponential). With p_transmit>0 that fraction of events is kept as the attenuated direct beam. mu(v) = absorption_coefficient*2200/v. Events for which the kernel cannot scatter (kinematically forbidden) are absorbed.

Geometry: box (xwidth,yheight,zdepth), cylinder (radius,yheight), hollow cylinder (+thickness) or sphere (radius only). All vectors (Q, targets, reciprocal vectors, atom positions) are in the component's local frame.

Uses the shared runtime share/mcvine-lib.h/.c.

Example: MCViNE_Phonon_IncoherentElastic(dw_core=0.0067, sigma_inc=10, sigma_abs=10, Vc=27.6, radius=0.01, yheight=0.05)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
dw_core	Å^2	Debye-Waller core u^2 (2W = dw_core*Q^2); <0: compute from dos	-1
dos	str	Phonon DOS file (2-column E[meV] g, or MCViNE IDF DOS) for dw_core<0	0
T	K	Temperature (for dw_core from DOS)	300
average_mass	amu	Average atomic mass (for dw_core from DOS)	0
sigma_inc	barn	Incoherent scattering cross section per unit cell	0
sigma_abs	barn	Absorption cross section per unit cell at 2200 m/s	0
Vc	Å^3	Unit cell volume	0
radius	m	Outer radius of a cylinder (with yheight) or of a sphere (yheight=0)	0
xwidth	m	Width of a box sample	0
yheight	m	Height of a box or cylinder sample	0
zdepth	m	Depth of a box sample	0

Name	Unit	Description	Default
thickness	m	Wall thickness of a hollow cylinder (0: filled)	0
pack	1	Packing factor (scales absorption and scattering coefficients)	1
p_transmit	1	Monte Carlo fraction of events kept as unscattered transmitted beam (0: always scatter)	0
order	1	Maximum number of scattering events per neutron (1: single scattering)	1

Links

- Component source code found in file `MCViNE_Phonon_IncoherentElastic.comp`.
- MCViNE documentation: https://mcvine.github.io

14.124. The `MCViNE_Phonon_IncoherentInelastic` McStas Component

One-phonon incoherent inelastic scattering from a phonon DOS (incoherent approximation), optional Ef focusing.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE phonon IncoherentInelastic / IncoherentInelastic_EnergyFocusing kernels; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/phonon/Incohe IncoherentInelastic_EnergyFocusing.cc)
- **Date:** 2026-09-29

Description

S_inc(Q,E) = exp(-2W) hbar^2Q^2/(2M) g(|E|)/|E| [n(E)+1 or n(E)] with the DOS g(E) normalised to 1, Debye-Waller factor computed from the DOS (MCViNE DWFromDOS), and the DOS preprocessed like MCViNE (resampled to >=500 points, parabolic low-energy fit, normalised). The final direction is uniform over 4pi; Ef is uniform in [Ei-Emax_dos, Ei+Emax_dos]. With dEf>0, Ef is restricted to Ef +/- dEf/2 (MCViNE IncoherentInelastic_EnergyFocusing). DOS file: 2 columns E [meV] g(E) (a '#...THz' comment switches the unit to THz) or an MCViNE IDF binary 'DOS' file. Multiphonon terms are not included (see Union IncoherentPhonon_process or NCrystal for those).

Transport (same as MCViNE HomogeneousNeutronScatterer): on the first passage the neutron is forced to scatter at a uniformly chosen depth, weighted by path length, attenuation exp(-(mu+sigma)x) and the scattering coefficient; on exit it is attenuated by exp(-(mu+sigma)L_out). With order>1 further scatterings are sampled analogically (truncated exponential). With p_transmit>0 that fraction of events is kept as the attenuated direct beam. mu(v) = absorption_coefficient*2200/v. Events for which the kernel cannot scatter (kinematically forbidden) are absorbed.

Geometry: box (xwidth,yheight,zdepth), cylinder (radius,yheight), hollow cylinder (+thickness) or sphere (radius only). All vectors (Q, targets, reciprocal vectors, atom positions) are in the component's local frame.

Uses the shared runtime share/mcvine-lib.h/.c.

Example: MCViNE_Phonon_IncoherentInelastic(dos="MCViNE/Debye_dos.dat", T=300, average_mass=50.94, sigma_inc=10.1, sigma_abs=10.1, Vc=27.6, radius=0.01, yheight=0.05)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
dos	str	Phonon DOS file	0
T	K	Temperature	300
average_mass	amu	Average atomic mass	0
Ef	meV	Final energy for energy focusing (used when dEf>0)	0
dEf	meV	Full width of the final-energy window; 0 disables focusing	0
sigma_inc	barn	Incoherent scattering cross section per unit cell	0
sigma_abs	barn	Absorption cross section per unit cell at 2200 m/s	0
Vc	$Å^3$	Unit cell volume	0
radius	m	Outer radius of a cylinder (with yheight) or of a sphere (yheight=0)	0
xwidth	m	Width of a box sample	0
yheight	m	Height of a box or cylinder sample	0
zdepth	m	Depth of a box sample	0
thickness	m	Wall thickness of a hollow cylinder (0: filled)	0
pack	1	Packing factor (scales absorption and scattering coefficients)	1
p_transmit	1	Monte Carlo fraction of events kept as unscattered transmitted beam (0: always scatter)	0

Name	Unit	Description	Default
order	1	Maximum number of scattering events per neutron (1: single scattering)	1

Links

- Component source code found in file `MCViNE_Phonon_IncoherentInelastic.comp`.
- MCViNE documentation: https://mcvine.github.io

14.125. The `MCViNE_Phonon_CoherentInelastic_PolyXtal` McStas Component

Coherent one-phonon scattering from a powder, using a full phonon dispersion (energies + polarizations) on a grid.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE phonon CoherentInelastic_PolyXtal kernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/phonon/Cohere
- **Date:** 2026-09-29

Description

Powder average of the coherent one-phonon cross section, computed from phonon energies and polarization vectors tabulated on a grid over one reciprocal cell (MCViNE IDF format, e.g. from phonopy via MCViNE tools). A random branch and a random Q vector in a cube are drawn until the event is kinematically allowed; phonon creation/annihilation, Bose factor and Debye-Waller factor included. Scattering coefficient: total coherent cross section / Vc.

Transport (same as MCViNE HomogeneousNeutronScatterer): on the first passage the neutron is forced to scatter at a uniformly chosen depth, weighted by path length, attenuation exp(-(mu+sigma)x) and the scattering coefficient; on exit it is attenuated by exp(-(mu+sigma)L_out). With order>1 further scatterings are sampled analogically (truncated exponential). With p_transmit>0 that fraction of events is kept as the attenuated direct beam. mu(v) = absorption_coefficient*2200/v. Events for which the kernel cannot scatter (kinematically forbidden) are absorbed.

Geometry: box (xwidth,yheight,zdepth), cylinder (radius,yheight), hollow cylinder (+thickness) or sphere (radius only). All vectors (Q, targets, reciprocal vectors, atom positions) are in the component's local frame.

Uses the shared runtime share/mcvine-lib.h/.c.

Example: MCViNE_Phonon_CoherentInelastic_PolyXtal(idf_dir="MCViNE/fcc_toy_phonons atoms="MCViNE/fcc_toy_atoms.dat", T=300, max_omega=45, radius=0.01, yheight=0.05)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
idf_dir	str	Directory with MCViNE IDF phonon files: Qgridinfo, Omega2, Polarizations[, DOS]	0
atoms	str	Atoms file: one row per atom 'x y z mass b_coh sigma_inc sigma_abs' ([Å] cartesian, [amu], [fm], [barn], [barn]), same order as in the IDF files	0
T	K	Temperature	300
dw_core	Å^2	Debye-Waller core; <0: computed from the DOS (idf_dir/DOS or dos)	-1
dos	str	Optional DOS file for the Debye-Waller factor (default: idf_dir/DOS)	0
Vc	Å^3	Unit cell volume; 0: (2pi)^3/|b1.(b2 x b3)| from Qgridinfo	0
max_omega	meV	Maximum phonon energy	50
min_omega	meV	Phonons below this energy are skipped	0.01
unbiased	1	0: MCViNE sampling (rejection + empirical accessible reciprocal volume, a few % high in tests). 1: single Q sample in the cube with its exact volume, unbiased	0
radius	m	Outer radius of a cylinder (with yheight) or of a sphere (yheight=0)	0
xwidth	m	Width of a box sample	0
yheight	m	Height of a box or cylinder sample	0
zdepth	m	Depth of a box sample	0
thickness	m	Wall thickness of a hollow cylinder (0: filled)	0
pack	1	Packing factor (scales absorption and scattering coefficients)	1
p_transmit	1	Monte Carlo fraction of events kept as unscattered transmitted beam (0: always scatter)	0
order	1	Maximum number of scattering events per neutron (1: single scattering)	1

Links

- Component source code found in file `MCViNE_Phonon_CoherentInelastic_PolyXtal.comp`.
- MCViNE documentation: https://mcvine.github.io

14.126. The `MCViNE_Phonon_CoherentInelastic_SingleXtal` McStas Component

Coherent one-phonon scattering from a single crystal, using a full phonon dispersion on a grid.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE phonon CoherentInelastic_SingleXtal kernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/phonon/Cohere
- **Date:** 2026-09-29

Description

Single-crystal coherent one-phonon scattering with dispersion and polarization vectors from MCViNE IDF files (periodic over the reciprocal cell b1,b2,b3 given in Qgridinfo; orient the crystal by rotating the component). For a final direction (4pi, or a cone towards a target when focus_r>0) and a random branch among those with Emin < 1.5 Ei, all final speeds solving omega(Q) = |Ei-Ef| are found (Ridders' method on nsteps subintervals of [0,2vi]); one is chosen and weighted by the delta-function Jacobian, |sum_d b_d/sqrt(M_d) exp(iQ.d)(Q.e_d)|^2, Bose and Debye-Waller factors.

Transport (same as MCViNE HomogeneousNeutronScatterer): on the first passage the neutron is forced to scatter at a uniformly chosen depth, weighted by path length, attenuation exp(-(mu+sigma)x) and the scattering coefficient; on exit it is attenuated by exp(-(mu+sigma)L_out). With order>1 further scatterings are sampled analogically (truncated exponential). With p_transmit>0 that fraction of events is kept as the attenuated direct beam. mu(v) = absorption_coefficient*2200/v. Events for which the kernel cannot scatter (kinematically forbidden) are absorbed.

Geometry: box (xwidth,yheight,zdepth), cylinder (radius,yheight), hollow cylinder (+thickness) or sphere (radius only). All vectors (Q, targets, reciprocal vectors, atom positions) are in the component's local frame.

Uses the shared runtime share/mcvine-lib.h/.c.

Example: MCViNE_Phonon_CoherentInelastic_SingleXtal(idf_dir="MCViNE/fcc_toy_phonons", atoms="MCViNE/fcc_toy_atoms.dat", T=300, xwidth=0.01, yheight=0.01, zdepth=0.01)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
idf_dir	str	Directory with MCViNE IDF phonon files: Qgridinfo, Omega2, Polarizations[, DOS]	0
atoms	str	Atoms file: one row per atom 'x y z mass b_coh sigma_inc sigma_abs' ([Å] cartesian, [amu], [fm], [barn], [barn]), same order as in the IDF files	0
T	K	Temperature	300
dw_core	$Å^2$	Debye-Waller core; <0: computed from the DOS (idf_dir/DOS or dos)	-1
dos	str	Optional DOS file for the Debye-Waller factor (default: idf_dir/DOS)	0
Vc	$Å^3$	Unit cell volume; 0: (2pi)^3/|b1.(b2 x b3)| from Qgridinfo	0
target_x	m	Target position (component frame) x	0
target_y	m	Target position y	0
target_z	m	Target position z	0
target_index	1	Relative index of a component to aim at (overrides target_x/y/z when non-zero)	0
focus_r	m	Radius of the target disk; 0: scatter into 4pi (MCViNE default)	0
deltaV_Jacobi	1	Relative velocity step for the numerical Jacobian	0.001
nsteps	1	Sub-intervals for the root search in vf	100
xacc	m/s	Root accuracy in vf	10
unbiased	1	0: MCViNE sampling (retries up to 100 direction/branch draws until omega(Q)=|Ei-Ef| has a solution, without correcting the weight: over-estimates when many directions have no solution). 1: single attempt, unbiased	0
radius	m	Outer radius of a cylinder (with yheight) or of a sphere (yheight=0)	0
xwidth	m	Width of a box sample	0
yheight	m	Height of a box or cylinder sample	0
zdepth	m	Depth of a box sample	0
thickness	m	Wall thickness of a hollow cylinder (0: filled)	0

Name	Unit	Description	Default
pack	1	Packing factor (scales absorption and scattering coefficients)	1
p_transmit	1	Monte Carlo fraction of events kept as unscattered transmitted beam (0: always scatter)	0
order	1	Maximum number of scattering events per neutron (1: single scattering)	1

Links

- Component source code found in file `MCViNE_Phonon_CoherentInelastic_SingleXtal.comp`.
- MCViNE documentation: https://mcvine.github.io

14.127. The `MCViNE_DGSSXRes` McStas Component

Resolution kernel for direct-geometry single-crystal spectrometers (aims at a pixel and TOF window).

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE DGSSXResKernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/DGSSXResKer
- **Date:** 2026-09-29

Description

Scatters towards a target disk (position target_x/y/z or target_index, radius focus_r) and picks the arrival time at the target uniformly in tof_at_target +/- dtof/2 (absolute neutron time, s). The final speed follows from the flight distance. Weight: solid_angle/4pi * dtof * dE/dt * vf/vi, as in MCViNE. Record ki/kf with monitors to build the resolution ellipsoid (similar in purpose to McStas' TOFRes_sample, with MCViNE's weighting).

Transport (same as MCViNE HomogeneousNeutronScatterer): on the first passage the neutron is forced to scatter at a uniformly chosen depth, weighted by path length, attenuation exp(-(mu+sigma)x) and the scattering coefficient; on exit it is attenuated by exp(-(mu+sigma)L_out). With order>1 further scatterings are sampled analogically (truncated exponential). With p_transmit>0 that fraction of events is kept as the attenuated direct beam. mu(v) = absorption_coefficient*2200/v. Events for which the kernel cannot scatter (kinematically forbidden) are absorbed.

Geometry: box (xwidth,yheight,zdepth), cylinder (radius,yheight), hollow cylinder (+thickness) or sphere (radius only). All vectors (Q, targets, reciprocal vectors, atom positions) are in the component's local frame.

Uses the shared runtime share/mcvine-lib.h/.c.

Example: MCViNE_DGSSXRes(target_index=2, focus_r=0.0125, tof_at_target=0.0052, dtof=1e-5, scattering_coefficient=10, xwidth=0.01, yheight=0.01, zdepth=0.01)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
target_x	m	Target position (component frame) x	0
target_y	m	Target position y	0
target_z	m	Target position z	0
target_index	1	Relative index of a component to aim at (overrides target_x/y/z when non-zero)	0
focus_r	m	Radius of the target disk	0.01
tof_at_target	s	Desired absolute time of arrival at the target	0.001
dtof	s	Width of the time window	1e-5
absorption_coefficient	m^{-1}	Absorption coefficient at 2200 m/s (MCViNE convention; scales as 1/v)	0
scattering_coefficient	m^{-1}	Scattering coefficient	0
sigma_abs	barn	Alternative: absorption cross section per unit cell at 2200 m/s (used when Vc>0)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0, coefficients are computed from sigma_abs, sigma_scat	0
radius	m	Outer radius of a cylinder (with yheight) or of a sphere (yheight=0)	0
xwidth	m	Width of a box sample	0
yheight	m	Height of a box or cylinder sample	0
zdepth	m	Depth of a box sample	0
thickness	m	Wall thickness of a hollow cylinder (0: filled)	0
pack	1	Packing factor (scales absorption and scattering coefficients)	1
p_transmit	1	Monte Carlo fraction of events kept as unscattered transmitted beam (0: always scatter)	0

Name	Unit	Description	Default
order	1	Maximum number of scattering events per neutron (1: single scattering)	1

Links

- Component source code found in file `MCViNE_DGSSXRes.comp`.
- MCViNE documentation: https://mcvine.github.io

14.128. The `MCViNE_SANS2D_ongrid` McStas Component

SANS from a 2D S(Qx,Qy) map on a grid (beam along z).

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE SANS2D_ongrid_Kernel (mcvine.acc); MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/acc (acc/kernels/SANS2D_ongrid.py)
- **Date:** 2026-09-29

Description

Qx, Qy are sampled uniformly over the grid range; the neutron gets v_perp -= Q*K2V and vz = sqrt(vi^2 - vx^2 - vy^2) (elastic). The weight is the bilinearly interpolated S(Qx,Qy) (no solid-angle Jacobian, as in mcvine.acc; use scattering_coefficient to set the absolute scale). File layout: first numeric row Qx values (nx, ascending), second row Qy values (ny), then ny rows of nx values (row index = Qy), i.e. the (y,x) image layout used by mcvine.acc.

Transport (same as MCViNE HomogeneousNeutronScatterer): on the first passage the neutron is forced to scatter at a uniformly chosen depth, weighted by path length, attenuation exp(-(mu+sigma)x) and the scattering coefficient; on exit it is attenuated by exp(-(mu+sigma)L_out). With order>1 further scatterings are sampled analogically (truncated exponential). With p_transmit>0 that fraction of events is kept as the attenuated direct beam. mu(v) = absorption_coefficient*2200/v. Events for which the kernel cannot scatter (kinematically forbidden) are absorbed.

Geometry: box (xwidth,yheight,zdepth), cylinder (radius,yheight), hollow cylinder (+thickness) or sphere (radius only). All vectors (Q, targets, reciprocal vectors, atom positions) are in the component's local frame.

Uses the shared runtime share/mcvine-lib.h/.c.

Example: MCViNE_SANS2D_ongrid(S_QxQy="MCViNE/SANS2D_example.dat", scattering_coefficient=1, xwidth=0.01, yheight=0.01, zdepth=0.001)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
S_QxQy	str	2D grid file (see description)	0
absorption_coefficient	m^{-1}	Absorption coefficient at 2200 m/s (MCViNE convention; scales as 1/v)	0
scattering_coefficient	m^{-1}	Scattering coefficient	0
sigma_abs	barn	Alternative: absorption cross section per unit cell at 2200 m/s (used when Vc>0)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0, coefficients are computed from sigma_abs, sigma_scat	0
radius	m	Outer radius of a cylinder (with yheight) or of a sphere (yheight=0)	0
xwidth	m	Width of a box sample	0
yheight	m	Height of a box or cylinder sample	0
zdepth	m	Depth of a box sample	0
thickness	m	Wall thickness of a hollow cylinder (0: filled)	0
pack	1	Packing factor (scales absorption and scattering coefficients)	1
p_transmit	1	Monte Carlo fraction of events kept as unscattered transmitted beam (0: always scatter)	0
order	1	Maximum number of scattering events per neutron (1: single scattering)	1

Links

- Component source code found in file `MCViNE_SANS2D_ongrid.comp`.
- MCViNE documentation: https://mcvine.github.io

14.129. Contributed MCViNE kernels as Union processes

Each of the MCViNE kernels above is also available as a process for the Union framework (chapter 12; see chapter B for the concepts). They are used like the other Union process components: they are named in the `process_string` of a `Union_make_material` component, may be combined with other processes in the same material, and the resulting

material can be assigned to any Union geometry. The parameters of each process are listed with the component below.

14.130. The `MCViNE_SQ_process` McStas Component

Union process: Elastic isotropic scatterer with structure factor S(|Q|) (expression or table).

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE SQkernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/SQkernel.cc (+ GridSQ, SQ_fromexpression))
- **Date:** 2026-09-29

Description

Elastic scattering with angular distribution given by S(|Q|), for liquids, glasses or powder-averaged diffuse scattering. S(Q) is either an expression in Q or a 2-column text file (Q [Å^-1], S), linearly interpolated (0 outside).

Functions are given as strings evaluated at run time (MCViNE uses fparser): + - * / ^ (or **), % , comparisons, && ||, if(c,a,b), sin cos tan asin acos atan sinh cosh tanh exp log log10 log2 sqrt abs floor ceil int sign cbrt, pow atan2 min max hypot fmod, constants pi and e.

Union process. Part of the Union components: define this process, collect it into a material with Union_make_material (absorption is set there with my_absorption, the absorption inverse penetration depth at 2200 m/s), assign the material to Union_box/Union_cylinder/Union_sphe geometries, and add a Union_master after them. Geometry, attenuation and multiple scattering are handled by Union_master; this component provides the MCViNE kernel: the scattering coefficient and the final-state sampling (weight). Isotropic process (powder/liquid-like); rotation is irrelevant. The kernel code is shared with the standalone component MCViNE_SQ (mcvine-lib.c). Uses share/mcvine-lib.h/.c and share/mcvine-union-lib.h/.c; the process type MCViNE is declared in share/union-lib.h and registered with Union_master in share/mcvine-union-lib.h.

Example: MCViNE_SQ_process(SQ="1+sin(Q*3)/(Q*3)", Qmin=0, Qmax=20, scattering_coefficient=10)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
SQ	str	S(Q) expression (ignored if SQ_file given)	"1"
SQ_file	str	2-column file Q S	0
Qmin	$Å^{-1}$	Lower Q bound	0
Qmax	$Å^{-1}$	Upper Q bound	100
scattering_coefficient	m^{-1}	Scattering coefficient (inverse penetration depth for scattering)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0 the coefficient is sigma_scat/Vc	0
packing_factor	1	Packing factor (scales the scattering coefficient)	1
interact_fraction	1	Union: fraction of interactions forced to this process (-1: by cross section)	-1
init	string	Deprecated and unused, accepted like on the other Union components (see Union_init).	""

Links

- Component source code found in file `MCViNE_SQ_process.comp`.
- MCViNE documentation: https://mcvine.github.io

14.131. The `MCViNE_SvQ_process` McStas Component

Union process: Elastic scatterer with a vector structure factor S(Qx,Qy,Qz) (expression or 3D grid).

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE SvQkernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/Sv (+ GridSvQ))
- **Date:** 2026-09-29

Description

Elastic scattering into a uniformly random direction, weighted by S(Q) with Q = ki - kf in the sample frame. S is either an expression in Qx,Qy,Qz or a 3D grid file: three

rows 'min max n' for Qx, Qy, Qz followed by nx*ny*nz values (Qx slowest, Qz fastest), trilinearly interpolated (0 outside).

Functions are given as strings evaluated at run time (MCViNE uses fparser): + - * / ^ (or **), % , comparisons, && ||, if(c,a,b), sin cos tan asin acos atan sinh cosh tanh exp log log10 log2 sqrt abs floor ceil int sign cbrt, pow atan2 min max hypot fmod, constants pi and e.

Union process. Part of the Union components: define this process, collect it into a material with Union_make_material (absorption is set there with my_absorption, the absorption inverse penetration depth at 2200 m/s), assign the material to Union_box/Union_cylinder/Union_sphe geometries, and add a Union_master after them. Geometry, attenuation and multiple scattering are handled by Union_master; this component provides the MCViNE kernel: the scattering coefficient and the final-state sampling (weight). Orientation: this process is anisotropic; its frame (Q vectors, reciprocal vectors, atom positions) follows the ROTATED placement of this component. The kernel code is shared with the standalone component MCViNE_SvQ (mcvine-lib.c). Uses share/mcvine-lib.h/.c and share/mcvine-union-lib.h/.c; the process type MCViNE is declared in share/union-lib.h and registered with Union_master in share/mcvine-union-lib.h.

Example: MCViNE_SvQ_process(SvQ="exp(-((Qx-1)^2+Qy^2+Qz^2)/0.01)", scattering_coefficient=10)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
SvQ	str	S(Qx,Qy,Qz) expression (ignored if SvQ_file given)	"1"
SvQ_file	str	3D grid file (see description)	0
scattering_coefficient	m^{-1}	Scattering coefficient (inverse penetration depth for scattering)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0 the coefficient is sigma_scat/Vc	0
packing_factor	1	Packing factor (scales the scattering coefficient)	1
interact_fraction	1	Union: fraction of interactions forced to this process (-1: by cross section)	-1
init	string	Deprecated and unused, accepted like on the other Union components (see Union_init).	""

Links

- Component source code found in file `MCViNE_SvQ_process.comp`.
- MCViNE documentation: https://mcvine.github.io

14.132. The `MCViNE_SQE_process` McStas Component

Union process: Isotropic S(Q,E) scatterer from an analytic expression or grid, with optional final-energy focusing.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE SQEkernel / SQE_EnergyFocusing_Kern (+ SQE_fromexpression); MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/SQ SQE_EnergyFocusing_Kernel.cc)
- **Date:** 2026-09-29

Description

Isotropic (powder / liquid) S(|Q|,E). E is sampled uniformly in [Emin, min(Ei,Emax)] and |Q| uniformly in the kinematically allowed part of [Qmin,Qmax]. S(Q,E) is an expression in Q [Å^-1] and E [meV], or a grid file in the Isotropic_Sqw format (first numeric row: q values; second: energy values; then nq rows of nw values), bilinearly interpolated. For grid data McStas' Isotropic_Sqw already exists; the new parts are the analytic S(Q,E) and the energy focusing: when dEf>0, E is restricted so that Ef lies in [Ef-dEf, Ef+dEf] (MCViNE SQE_EnergyFocusing_Kernel).

Functions are given as strings evaluated at run time (MCViNE uses fparser): + - * / ^ (or **), % , comparisons, && ||, if(c,a,b), sin cos tan asin acos atan sinh cosh tanh exp log log10 log2 sqrt abs floor ceil int sign cbrt, pow atan2 min max hypot fmod, constants pi and e.

Union process. Part of the Union components: define this process, collect it into a material with Union_make_material (absorption is set there with my_absorption, the absorption inverse penetration depth at 2200 m/s), assign the material to Union_box/Union_cylinder/ geometries, and add a Union_master after them. Geometry, attenuation and multiple scattering are handled by Union_master; this component provides the MCViNE kernel: the scattering coefficient and the final-state sampling (weight). Isotropic process (powder/liquid-like); rotation is irrelevant. The kernel code is shared with the standalone component MCViNE_SQE (mcvine-lib.c). Uses share/mcvine-lib.h/.c and share/mcvine-union-lib.h/.c; the process type MCViNE is declared in share/union-lib.h and registered with Union_master in share/mcvine-union-lib.h.

Example: MCViNE_SQE_process(SQE="exp(-(E-10)^2/2)*Q^2*exp(-Q^2/4)", Qmin=0, Qmax=10, Emin=-5, Emax=30, sigma_scat=5, Vc=50)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
SQE	str	S(Q,E) expression (ignored if SQE_file given)	"1"
SQE_file	str	S(q,w) grid file (Isotropic_Sqw layout)	0
Qmin	$Å^{-1}$	Lower Q bound	0
Qmax	$Å^{-1}$	Upper Q bound	10
Emin	meV	Lower energy transfer bound	-10
Emax	meV	Upper energy transfer bound	10
Ef	meV	Final-energy focusing centre (used when dEf>0)	0
dEf	meV	Final-energy focusing half range; 0 disables focusing	0
scattering_coefficient	m^{-1}	Scattering coefficient (inverse penetration depth for scattering)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0 the coefficient is sigma_scat/Vc	0
packing_factor	1	Packing factor (scales the scattering coefficient)	1
interact_fraction	1	Union: fraction of interactions forced to this process (-1: by cross section)	-1
init	string	Deprecated and unused, accepted like on the other Union components (see Union_init).	""

Links

- Component source code found in file `MCViNE_SQE_process.comp`.
- MCViNE documentation: https://mcvine.github.io

14.133. The `MCViNE_E_Q_process` McStas Component

Union process: Isotropic dispersion: S(Q,E) = S(Q) delta(E - E(Q)) with analytic E(Q), S(Q).

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE E_Q_Kernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/E_
- **Date:** 2026-09-29

Description

Powder/liquid-like excitation with dispersion E(|Q|) given as an expression in Q [Å^-1] returning meV, and intensity S(Q). |Q| is sampled uniformly in [Qmin,Qmax] (up to 100 attempts to find a kinematically allowed Q).

Functions are given as strings evaluated at run time (MCViNE uses fparser): + - * / ^ (or **), % , comparisons, && ||, if(c,a,b), sin cos tan asin acos atan sinh cosh tanh exp log log10 log2 sqrt abs floor ceil int sign cbrt, pow atan2 min max hypot fmod, constants pi and e.

Union process. Part of the Union components: define this process, collect it into a material with Union_make_material (absorption is set there with my_absorption, the absorption inverse penetration depth at 2200 m/s), assign the material to Union_box/Union_cylinder/ geometries, and add a Union_master after them. Geometry, attenuation and multiple scattering are handled by Union_master; this component provides the MCViNE kernel: the scattering coefficient and the final-state sampling (weight). Isotropic process (powder/liquid-like); rotation is irrelevant. The kernel code is shared with the standalone component MCViNE_E_Q (mcvine-lib.c). Uses share/mcvine-lib.h/.c and share/mcvine-union-lib.h/.c; the process type MCViNE is declared in share/union-lib.h and registered with Union_master in share/mcvine-union-lib.h.

Example: MCViNE_E_Q_process(E_Q="20*sin(Q*1.5)^2", S_Q="1", Qmin=0, Qmax=10, scattering_coefficient=10)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
E_Q	str	E(Q) expression, variable Q [Å^-1], result [meV]	"10"
S_Q	str	S(Q) expression, variable Q	"1"
Qmin	$Å^{-1}$	Lower Q bound	0
Qmax	$Å^{-1}$	Upper Q bound	10
unbiased	1	0: MCViNE sampling (up to 100 retries, weight/attempts; slightly biased when part of [Qmin,Qmax] is forbidden). 1: single attempt, unbiased	0

Name	Unit	Description	Default
scattering_coefficient	m^{-1}	Scattering coefficient (inverse penetration depth for scattering)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0 the coefficient is sigma_scat/Vc	0
packing_factor	1	Packing factor (scales the scattering coefficient)	1
interact_fraction	1	Union: fraction of interactions forced to this process (-1: by cross section)	-1
init	string	Deprecated and unused, accepted like on the other Union components (see Union_init).	""

Links

- Component source code found in file `MCViNE_E_Q_process.comp`.
- MCViNE documentation: https://mcvine.github.io

14.134. The `MCViNE_E_vQ_process` McStas Component

Union process: Single-crystal dispersion: S(Q,E) = S(Q) delta(E - E(Q)) with analytic E(Qx,Qy,Qz).

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE E_vQ_Kernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/E_vQ_Kernel.
- **Date:** 2026-09-29

Description

Excitation with dispersion E(Qx,Qy,Qz) [meV] and intensity S(Qx,Qy,Qz), given as expressions of the Q vector in the sample (component) frame [Å^-1]. For a random final direction, all solutions kf of Ei - Ef(kf) = E(ki - kf) with Ef in [Ei-Emax, Ei] are found (nsteps sub-intervals + Ridders' method) and one is picked; the weight contains the Jacobian of the delta function.

Functions are given as strings evaluated at run time (MCViNE uses fparser): + - * / ^ (or **), % , comparisons, && ||, if(c,a,b), sin cos tan asin acos atan sinh cosh tanh exp

log log10 log2 sqrt abs floor ceil int sign cbrt, pow atan2 min max hypot fmod, constants pi and e.

Union process. Part of the Union components: define this process, collect it into a material with Union_make_material (absorption is set there with my_absorption, the absorption inverse penetration depth at 2200 m/s), assign the material to Union_box/Union_cylinder/ geometries, and add a Union_master after them. Geometry, attenuation and multiple scattering are handled by Union_master; this component provides the MCViNE kernel: the scattering coefficient and the final-state sampling (weight). Orientation: this process is anisotropic; its frame (Q vectors, reciprocal vectors, atom positions) follows the ROTATED placement of this component. The kernel code is shared with the standalone component MCViNE_E_vQ (mcvine-lib.c). Uses share/mcvine-lib.h/.c and share/mcvine-union-lib.h/.c; the process type MCViNE is declared in share/union-lib.h and registered with Union_master in share/mcvine-union-lib.h.

Example: MCViNE_E_vQ_process(E_Q="20*(sin(Qx*1.57)^2+sin(Qy*1.57)^2+sin(Qz*1.57)^ S_Q="1", Emax=60, scattering_coefficient=10)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
E_Q	str	E(Qx,Qy,Qz) expression [meV]	"10"
S_Q	str	S(Qx,Qy,Qz) expression	"1"
Emax	meV	Maximum energy transfer considered	10
nsteps	1	Number of sub-intervals for the kf root search (MCViNE: 1000)	1000
xacc	$Å^{-1}$	Root accuracy in kf (MCViNE: 1e-3)	1e-3
scattering_coefficient	m^{-1}	Scattering coefficient (inverse penetration depth for scattering)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0 the coefficient is sigma_scat/Vc	0
packing_factor	1	Packing factor (scales the scattering coefficient)	1
interact_fraction	1	Union: fraction of interactions forced to this process (-1: by cross section)	-1
init	string	Deprecated and unused, accepted like on the other Union components (see Union_init).	""

Links

- Component source code found in file `MCViNE_E_vQ_process.comp`.

- MCViNE documentation: https://mcvine.github.io

14.135. The `MCViNE_Broadened_E_Q_process` McStas Component

Union process: Isotropic dispersion E(Q) broadened by a Gaussian of Q-dependent width.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE Broadened_E_Q_Kernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/Broadened_E_
- **Date:** 2026-09-29

Description

S(Q,E) = S(Q) G(E - E(Q); sigma(Q)) with G a normalised Gaussian of standard deviation sigma(Q). Q is sampled uniformly in [Qmin,Qmax]; the energy offset is sampled from the line shape (up to 100 attempts). Neutrons with Ei below min(E(Q)-3w(Q)) do not scatter.

Functions are given as strings evaluated at run time (MCViNE uses fparser): + - * / ^ (or **), % , comparisons, && ||, if(c,a,b), sin cos tan asin acos atan sinh cosh tanh exp log log10 log2 sqrt abs floor ceil int sign cbrt, pow atan2 min max hypot fmod, constants pi and e.

Union process. Part of the Union components: define this process, collect it into a material with Union_make_material (absorption is set there with my_absorption, the absorption inverse penetration depth at 2200 m/s), assign the material to Union_box/Union_cylinder/Union_sphe geometries, and add a Union_master after them. Geometry, attenuation and multiple scattering are handled by Union_master; this component provides the MCViNE kernel: the scattering coefficient and the final-state sampling (weight). Isotropic process (powder/liquid-like); rotation is irrelevant. The kernel code is shared with the standalone component MCViNE_Broadened_E_Q (mcvine-lib.c). Uses share/mcvine-lib.h/.c and share/mcvine-union-lib.h/.c; the process type MCViNE is declared in share/union-lib.h and registered with Union_master in share/mcvine-union-lib.h.

Example: MCViNE_Broadened_E_Q_process(E_Q="20*sin(Q*1.5)^2", S_Q="1", sigma_Q="0.5", Qmin=0, Qmax=10, scattering_coefficient=10)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
E_Q	str	E(Q) expression [meV], variable Q [Å^-1]	"10"
S_Q	str	S(Q) expression	"1"
sigma_Q	str	Gaussian sigma(Q) [meV]	"1"
Qmin	$Å^{-1}$	Lower Q bound	0
Qmax	$Å^{-1}$	Upper Q bound	10
unbiased	1	0: MCViNE sampling (retries without acceptance correction: over-weights when part of the Q range is forbidden). 1: single attempt, unbiased	0
scattering_coefficient	m^{-1}	Scattering coefficient (inverse penetration depth for scattering)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0 the coefficient is sigma_scat/Vc	0
packing_factor	1	Packing factor (scales the scattering coefficient)	1
interact_fraction	1	Union: fraction of interactions forced to this process (-1: by cross section)	-1
init	string	Deprecated and unused, accepted like on the other Union components (see Union_init).	""

Links

- Component source code found in file `MCViNE_Broadened_E_Q_process.comp`.
- MCViNE documentation: https://mcvine.github.io

14.136. The `MCViNE_LorentzianBroadened_E_Q_process` McStas Component

Union process: Isotropic dispersion E(Q) broadened by a Lorentzian of Q-dependent width.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE LorentzianBroadened_E_Q_Kernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/Lo
- **Date:** 2026-09-29

Description

S(Q,E) = S(Q) L(E - E(Q); gamma(Q)) with L a Lorentzian of half width at half maximum gamma(Q). Q is sampled uniformly in [Qmin,Qmax]; the energy offset is sampled from the line shape (up to 100 attempts). Neutrons with Ei below min(E(Q)-3w(Q)) do not scatter.

Functions are given as strings evaluated at run time (MCViNE uses fparser): + - * / ^ (or **), % , comparisons, && ||, if(c,a,b), sin cos tan asin acos atan sinh cosh tanh exp log log10 log2 sqrt abs floor ceil int sign cbrt, pow atan2 min max hypot fmod, constants pi and e.

Union process. Part of the Union components: define this process, collect it into a material with Union_make_material (absorption is set there with my_absorption, the absorption inverse penetration depth at 2200 m/s), assign the material to Union_box/Union_cylinder/Union_sphe geometries, and add a Union_master after them. Geometry, attenuation and multiple scattering are handled by Union_master; this component provides the MCViNE kernel: the scattering coefficient and the final-state sampling (weight). Isotropic process (powder/liquid-like); rotation is irrelevant. The kernel code is shared with the standalone component MCViNE_LorentzianBroadened_E_Q (mcvine-lib.c). Uses share/mcvine-lib.h/.c and share/mcvine-union-lib.h/.c; the process type MCViNE is declared in share/union-lib.h and registered with Union_master in share/mcvine-union-lib.h.

Example: MCViNE_LorentzianBroadened_E_Q_process(E_Q="20*sin(Q*1.5)^2", S_Q="1", gamma_Q="0.5", Qmin=0, Qmax=10, scattering_coefficient=10)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
E_Q	str	E(Q) expression [meV], variable Q [Å^-1]	"10"
S_Q	str	S(Q) expression	"1"
gamma_Q	str	Lorentzian HWHM gamma(Q) [meV]	"1"
Qmin	$Å^{-1}$	Lower Q bound	0
Qmax	$Å^{-1}$	Upper Q bound	10
unbiased	1	0: MCViNE sampling (retries without acceptance correction: over-weights when part of the Q range is forbidden). 1: single attempt, unbiased	0
scattering_coefficient	m^{-1}	Scattering coefficient (inverse penetration depth for scattering)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0 the coefficient is sigma_scat/Vc	0

Name	Unit	Description	Default
packing_factor	1	Packing factor (scales the scattering coefficient)	1
interact_fraction	1	Union: fraction of interactions forced to this process (-1: by cross section)	-1
init	string	Deprecated and unused, accepted like on the other Union components (see Union_init).	""

Links

- Component source code found in file `MCViNE_LorentzianBroadened_E_Q_process.comp`.
- MCViNE documentation: https://mcvine.github.io

14.137. The `MCViNE_ConstantEnergyTransfer_process` McStas Component

Union process: Isotropic scatterer with a fixed energy transfer.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE ConstantEnergyTransferKernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/Co
- **Date:** 2026-09-29

Description

Every scattered neutron loses exactly the energy E (E<0: gains) and leaves in a uniformly random direction over 4pi. Useful for testing and resolution studies.

Union process. Part of the Union components: define this process, collect it into a material with Union_make_material (absorption is set there with my_absorption, the absorption inverse penetration depth at 2200 m/s), assign the material to Union_box/Union_cylinder/ geometries, and add a Union_master after them. Geometry, attenuation and multiple scattering are handled by Union_master; this component provides the MCViNE kernel: the scattering coefficient and the final-state sampling (weight). Isotropic process (powder/liquid-like); rotation is irrelevant. The kernel code is shared with the standalone component MCViNE_ConstantEnergyTransfer (mcvine-lib.c). Uses share/mcvine-lib.h/.c and share/mcvine-union-lib.h/.c; the process type MCViNE is declared in share/union-lib.h and registered with Union_master in share/mcvine-union-lib.h.

Example: MCViNE_ConstantEnergyTransfer_process(E=10, scattering_coefficient=10)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
E	meV	Energy transfer Ei-Ef	0
scattering_coefficient	m^{-1}	Scattering coefficient (inverse penetration depth for scattering)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0 the coefficient is sigma_scat/Vc	0
packing_factor	1	Packing factor (scales the scattering coefficient)	1
interact_fraction	1	Union: fraction of interactions forced to this process (-1: by cross section)	-1
init	string	Deprecated and unused, accepted like on the other Union components (see Union_init).	""

Links

- Component source code found in file `MCViNE_ConstantEnergyTransfer_process.comp`.
- MCViNE documentation: https://mcvine.github.io

14.138. The `MCViNE_ConstantQE_process` McStas Component

Union process: Scatterer with fixed momentum transfer |Q| and energy transfer E.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE ConstantQEKernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/ConstantQEKe
- **Date:** 2026-09-29

Description

Scattered neutrons have energy transfer E and |Q| fixed; the azimuth around ki is random. S(Q,E) = delta(|Q|-Q0) delta(E-E0) (unnormalised, as in MCViNE).

Union process. Part of the Union components: define this process, collect it into a material with Union_make_material (absorption is set there with my_absorption, the

absorption inverse penetration depth at 2200 m/s), assign the material to Union_box/Union_cylinder/ geometries, and add a Union_master after them. Geometry, attenuation and multiple scattering are handled by Union_master; this component provides the MCViNE kernel: the scattering coefficient and the final-state sampling (weight). Isotropic process (powder/liquid-like); rotation is irrelevant. The kernel code is shared with the standalone component MCViNE_ConstantQE (mcvine-lib.c). Uses share/mcvine-lib.h/.c and share/mcvine-union-lib.h/.c; the process type MCViNE is declared in share/union-lib.h and registered with Union_master in share/mcvine-union-lib.h.

Example: MCViNE_ConstantQE_process(Q=3, E=20, scattering_coefficient=10)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
Q	$Å^{-1}$	Momentum transfer |Q|	1
E	meV	Energy transfer	0
scattering_coefficient	m^{-1}	Scattering coefficient (inverse penetration depth for scattering)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0 the coefficient is sigma_scat/Vc	0
packing_factor	1	Packing factor (scales the scattering coefficient)	1
interact_fraction	1	Union: fraction of interactions forced to this process (-1: by cross section)	-1
init	string	Deprecated and unused, accepted like on the other Union components (see Union_init).	""

Links

- Component source code found in file `MCViNE_ConstantQE_process.comp`.
- MCViNE documentation: https://mcvine.github.io

14.139. The `MCViNE_ConstantvQE_process` McStas Component

Union process: Scatterer with a fixed momentum-transfer vector Q, weighted by a Gaussian in E.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE ConstantvQEKernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/ConstantvQEK
- **Date:** 2026-09-29

Description

kf = ki - Q (Q vector fixed in the sample frame); the event weight is multiplied by exp(-(E-E0)^2/(2 dE^2)) where E = Ei-Ef. Used for single-crystal resolution studies.

Union process. Part of the Union components: define this process, collect it into a material with Union_make_material (absorption is set there with my_absorption, the absorption inverse penetration depth at 2200 m/s), assign the material to Union_box/Union_cylinder/Union_sphe geometries, and add a Union_master after them. Geometry, attenuation and multiple scattering are handled by Union_master; this component provides the MCViNE kernel: the scattering coefficient and the final-state sampling (weight). Orientation: this process is anisotropic; its frame (Q vectors, reciprocal vectors, atom positions) follows the ROTATED placement of this component. The kernel code is shared with the standalone component MCViNE_ConstantvQE (mcvine-lib.c). Uses share/mcvine-lib.h/.c and share/mcvine-union-lib.h/.c; the process type MCViNE is declared in share/union-lib.h and registered with Union_master in share/mcvine-union-lib.h.

Example: MCViNE_ConstantvQE_process(Qx=1, Qy=0, Qz=0.5, E=10, dE=1, scattering_coefficient=10)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
Qx	$Å^{-1}$	Q vector x	0
Qy	$Å^{-1}$	Q vector y	0
Qz	$Å^{-1}$	Q vector z	0
E	meV	Nominal energy transfer	0
dE	meV	Gaussian sigma of the energy weight	1
scattering_coefficient	m^{-1}	Scattering coefficient (inverse penetration depth for scattering)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0 the coefficient is sigma_scat/Vc	0
packing_factor	1	Packing factor (scales the scattering coefficient)	1

Name	Unit	Description	Default
interact_fraction	1	Union: fraction of interactions forced to this process (-1: by cross section)	-1
init	string	Deprecated and unused, accepted like on the other Union components (see Union_init).	""

Links

- Component source code found in file `MCViNE_ConstantvQE_process.comp`.
- MCViNE documentation: https://mcvine.github.io

14.140. The `MCViNE_Phonon_IncoherentElastic_process` McStas Component

Union process: Incoherent elastic scattering with Debye-Waller factor exp(-2W), 2W = dw_core*Q^2.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE phonon IncoherentElastic kernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/ph
- **Date:** 2026-09-29

Description

Isotropic incoherent elastic scattering damped by the Debye-Waller factor. dw_core [Å^2] is given directly, or (dw_core<0) computed from a phonon DOS, temperature and average atomic mass exactly as MCViNE's DWFromDOS. McStas' Incoherent has no Debye-Waller factor.

Union process. Part of the Union components: define this process, collect it into a material with Union_make_material (absorption is set there with my_absorption, the absorption inverse penetration depth at 2200 m/s), assign the material to Union_box/Union_cylinder/ geometries, and add a Union_master after them. Geometry, attenuation and multiple scattering are handled by Union_master; this component provides the MCViNE kernel: the scattering coefficient and the final-state sampling (weight). Isotropic process (powder/liquid-like); rotation is irrelevant. The kernel code is shared with the standalone component MCViNE_Phonon_IncoherentElastic (mcvine-lib.c). Uses share/mcvine-lib.h/.c and share/mcvine-union-lib.h/.c; the process type MCViNE is declared in share/union-lib.h and registered with Union_master in share/mcvine-union-lib.h.

Example: MCViNE_Phonon_IncoherentElastic_process(dw_core=0.0067, sigma_inc=10, Vc=27.6)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
dw_core	Å^2	Debye-Waller core u^2 (2W = dw_core*Q^2); <0: compute from dos	-1
dos	str	Phonon DOS file (2-column E[meV] g, or MCViNE IDF DOS) for dw_core<0	0
T	K	Temperature (for dw_core from DOS)	300
average_mass	amu	Average atomic mass (for dw_core from DOS)	0
sigma_inc	barn	Incoherent scattering cross section per unit cell	0
Vc	Å^3	Unit cell volume	0
packing_factor	1	Packing factor (scales the scattering coefficient)	1
interact_fraction	1	Union: fraction of interactions forced to this process (-1: by cross section)	-1
init	string	Deprecated and unused, accepted like on the other Union components (see Union_init).	""

Links

- Component source code found in file `MCViNE_Phonon_IncoherentElastic_process.comp`.
- MCViNE documentation: https://mcvine.github.io

14.141. The `MCViNE_Phonon_IncoherentInelastic_process` McStas Component

Union process: One-phonon incoherent inelastic scattering from a phonon DOS (incoherent approximation), optional Ef focusing.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE phonon IncoherentInelastic / IncoherentInelastic_EnergyFocusing kernels; MCViNE by J. Y. Y. Lin et al.)

- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/ph IncoherentInelastic_EnergyFocusing.cc)
- **Date:** 2026-09-29

Description

S_inc(Q,E) = exp(-2W) hbar^2Q^2/(2M) g(|E|)/|E| [n(E)+1 or n(E)] with the DOS g(E) normalised to 1, Debye-Waller factor computed from the DOS (MCViNE DWFromDOS), and the DOS preprocessed like MCViNE (resampled to >=500 points, parabolic low-energy fit, normalised). The final direction is uniform over 4pi; Ef is uniform in [Ei-Emax_dos, Ei+Emax_dos]. With dEf>0, Ef is restricted to Ef +/- dEf/2 (MCViNE IncoherentInelastic_EnergyFocusing). DOS file: 2 columns E [meV] g(E) (a '#...THz' comment switches the unit to THz) or an MCViNE IDF binary 'DOS' file. Multiphonon terms are not included (see Union IncoherentPhonon_process or NCrystal for those).

Union process. Part of the Union components: define this process, collect it into a material with Union_make_material (absorption is set there with my_absorption, the absorption inverse penetration depth at 2200 m/s), assign the material to Union_box/Union_cylinder/ geometries, and add a Union_master after them. Geometry, attenuation and multiple scattering are handled by Union_master; this component provides the MCViNE kernel: the scattering coefficient and the final-state sampling (weight). Isotropic process (powder/liquid-like); rotation is irrelevant. The kernel code is shared with the standalone component MCViNE_Phonon_IncoherentInelastic (mcvine-lib.c). Uses share/mcvine-lib.h/.c and share/mcvine-union-lib.h/.c; the process type MCViNE is declared in share/union-lib.h and registered with Union_master in share/mcvine-union-lib.h.

Example: MCViNE_Phonon_IncoherentInelastic_process(dos="MCViNE/Debye_dos.dat", T=300, average_mass=50.94, sigma_inc=10.1, Vc=27.6)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
dos	str	Phonon DOS file	0
T	K	Temperature	300
average_mass	amu	Average atomic mass	0
Ef	meV	Final energy for energy focusing (used when dEf>0)	0
dEf	meV	Full width of the final-energy window; 0 disables focusing	0
sigma_inc	barn	Incoherent scattering cross section per unit cell	0
Vc	$Å^3$	Unit cell volume	0

Name	Unit	Description	Default
packing_factor	1	Packing factor (scales the scattering coefficient)	1
interact_fraction	1	Union: fraction of interactions forced to this process (-1: by cross section)	-1
init	string	Deprecated and unused, accepted like on the other Union components (see Union_init).	""

Links

- Component source code found in file `MCViNE_Phonon_IncoherentInelastic_process.comp`.
- MCViNE documentation: https://mcvine.github.io

14.142. The `MCViNE_Phonon_CoherentInelastic_PolyXtal_process` McStas Component

Union process: Coherent one-phonon scattering from a powder, using a full phonon dispersion (energies + polarizations) on a grid.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE phonon CoherentInelastic_PolyXtal kernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/phonon/Cohere
- **Date:** 2026-09-29

Description

Powder average of the coherent one-phonon cross section, computed from phonon energies and polarization vectors tabulated on a grid over one reciprocal cell (MCViNE IDF format, e.g. from phonopy via MCViNE tools). A random branch and a random Q vector in a cube are drawn until the event is kinematically allowed; phonon creation/annihilation, Bose factor and Debye-Waller factor included. Scattering coefficient: total coherent cross section / Vc.

Union process. Part of the Union components: define this process, collect it into a material with Union_make_material (absorption is set there with my_absorption, the absorption inverse penetration depth at 2200 m/s), assign the material to Union_box/Union_cylinder/Union_sphe geometries, and add a Union_master after them. Geometry, attenuation and multiple

scattering are handled by Union_master; this component provides the MCViNE kernel: the scattering coefficient and the final-state sampling (weight). Isotropic process (powder/liquid-like); rotation is irrelevant. The kernel code is shared with the standalone component MCViNE_Phonon_CoherentInelastic_PolyXtal (mcvine-lib.c). Uses share/mcvine-lib.h/.c and share/mcvine-union-lib.h/.c; the process type MCViNE is declared in share/union-lib.h and registered with Union_master in share/mcvine-union-lib.h.

Example: MCViNE_Phonon_CoherentInelastic_PolyXtal_process(idf_dir="MCViNE/fcc_toy_
atoms="MCViNE/fcc_toy_atoms.dat", T=300, max_omega=45)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
idf_dir	str	Directory with MCViNE IDF phonon files: Qgridinfo, Omega2, Polarizations[, DOS]	0
atoms	str	Atoms file: one row per atom 'x y z mass b_coh sigma_inc sigma_abs' ([Å] cartesian, [amu], [fm], [barn], [barn]), same order as in the IDF files	0
T	K	Temperature	300
dw_core	$Å^2$	Debye-Waller core; <0: computed from the DOS (idf_dir/DOS or dos)	-1
dos	str	Optional DOS file for the Debye-Waller factor (default: idf_dir/DOS)	0
Vc	$Å^3$	Unit cell volume; 0: (2pi)^3/|b1.(b2 x b3)| from Qgridinfo	0
max_omega	meV	Maximum phonon energy	50
min_omega	meV	Phonons below this energy are skipped	0.01
unbiased	1	0: MCViNE sampling (rejection + empirical accessible reciprocal volume, a few % high in tests). 1: single Q sample in the cube with its exact volume, unbiased	0
packing_factor	1	Packing factor (scales the scattering coefficient)	1
interact_fraction	1	Union: fraction of interactions forced to this process (-1: by cross section)	-1
init	string	Deprecated and unused, accepted like on the other Union components (see Union_init).	""

Links

- Component source code found in file `MCViNE_Phonon_CoherentInelastic_PolyXtal_process.comp`.
- MCViNE documentation: https://mcvine.github.io

14.143. The MCViNE_Phonon_CoherentInelastic_SingleXtal_process McStas Component

Union process: Coherent one-phonon scattering from a single crystal, using a full phonon dispersion on a grid.

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE phonon CoherentInelastic_SingleXtal kernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/phonon/Cohere
- **Date:** 2026-09-29

Description

Single-crystal coherent one-phonon scattering with dispersion and polarization vectors from MCViNE IDF files (periodic over the reciprocal cell b1,b2,b3 given in Qgridinfo; orient the crystal by rotating the component). For a final direction (4pi, or a cone towards a target when focus_r>0) and a random branch among those with Emin < 1.5 Ei, all final speeds solving omega(Q) = |Ei-Ef| are found (Ridders' method on nsteps subintervals of [0,2vi]); one is chosen and weighted by the delta-function Jacobian, |sum_d b_d/sqrt(M_d) exp(iQ.d)(Q.e_d)|^2, Bose and Debye-Waller factors.

Union process. Part of the Union components: define this process, collect it into a material with Union_make_material (absorption is set there with my_absorption, the absorption inverse penetration depth at 2200 m/s), assign the material to Union_box/Union_cylinder/Union_sphe geometries, and add a Union_master after them. Geometry, attenuation and multiple scattering are handled by Union_master; this component provides the MCViNE kernel: the scattering coefficient and the final-state sampling (weight). Orientation: this process is anisotropic; its frame (Q vectors, reciprocal vectors, atom positions) follows the ROTATED placement of this component. The kernel code is shared with the standalone component MCViNE_Phonon_CoherentInelastic_SingleXtal (mcvine-lib.c). Uses share/mcvine-lib.h/.c and share/mcvine-union-lib.h/.c; the process type MCViNE is declared in share/union-lib.h and registered with Union_master in share/mcvine-union-lib.h.

Example: MCViNE_Phonon_CoherentInelastic_SingleXtal_process(idf_dir="MCViNE/fcc_toy_phonons", atoms="MCViNE/fcc_toy_atoms.dat", T=300)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
idf_dir	str	Directory with MCViNE IDF phonon files: Qgridinfo, Omega2, Polarizations[, DOS]	0
atoms	str	Atoms file: one row per atom 'x y z mass b_coh sigma_inc sigma_abs' ([Å] cartesian, [amu], [fm], [barn], [barn]), same order as in the IDF files	0
T	K	Temperature	300
dw_core	$Å^2$	Debye-Waller core; <0: computed from the DOS (idf_dir/DOS or dos)	-1
dos	str	Optional DOS file for the Debye-Waller factor (default: idf_dir/DOS)	0
Vc	$Å^3$	Unit cell volume; 0: (2pi)^3/|b1.(b2 x b3)| from Qgridinfo	0
deltaV_Jacobi	1	Relative velocity step for the numerical Jacobian	0.001
nsteps	1	Sub-intervals for the root search in vf	100
xacc	m/s	Root accuracy in vf	10
unbiased	1	0: MCViNE sampling (retries up to 100 direction/branch draws until omega(Q)=|Ei-Ef| has a solution, without correcting the weight: over-estimates when many directions have no solution). 1: single attempt, unbiased	0
packing_factor	1	Packing factor (scales the scattering coefficient)	1
interact_fraction	1	Union: fraction of interactions forced to this process (-1: by cross section)	-1
init	string	Deprecated and unused, accepted like on the other Union components (see Union_init).	""

Links

- Component source code found in file `MCViNE_Phonon_CoherentInelastic_SingleXtal_process`
- MCViNE documentation: https://mcvine.github.io

14.144. The `MCViNE_DGSSXRes_process` McStas Component

Union process: Resolution kernel for direct-geometry single-crystal spectrometers (aims at a pixel and TOF window).

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE DGSSXResKernel; MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/mcvine (mccomponents/lib/kernels/sample/DGSSXResKer
- **Date:** 2026-09-29

Description

Scatters towards a target disk (position target_x/y/z or target_index, radius focus_r) and picks the arrival time at the target uniformly in tof_at_target +/- dtof/2 (absolute neutron time, s). The final speed follows from the flight distance. Weight: solid_angle/4pi * dtof * dE/dt * vf/vi, as in MCViNE. Record ki/kf with monitors to build the resolution ellipsoid (similar in purpose to McStas' TOFRes_sample, with MCViNE's weighting).

Union process. Part of the Union components: define this process, collect it into a material with Union_make_material (absorption is set there with my_absorption, the absorption inverse penetration depth at 2200 m/s), assign the material to Union_box/Union_cylinder/Union_sphe geometries, and add a Union_master after them. Geometry, attenuation and multiple scattering are handled by Union_master; this component provides the MCViNE kernel: the scattering coefficient and the final-state sampling (weight). Isotropic process (powder/liquid-like); rotation is irrelevant. The target is the Union focusing of the geometry (target_index / target_x.. and focus_r / focus_xw,focus_xh / focus_aw,focus_ah on Union_box etc.). The kernel code is shared with the standalone component MCViNE_DGSSXRes (mcvine-lib.c). Uses share/mcvine-lib.h/.c and share/mcvine-union-lib.h/.c; the process type MCViNE is declared in share/union-lib.h and registered with Union_master in share/mcvine-union-lib.h.

Example: MCViNE_DGSSXRes_process(focus_r=0.0125, tof_at_target=0.0052, dtof=1e-5, scattering_coefficient=10)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
tof_at_target	s	Desired absolute time of arrival at the target	0.001
dtof	s	Width of the time window	1e-5

Name	Unit	Description	Default
scattering_coefficient	m^{-1}	Scattering coefficient (inverse penetration depth for scattering)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0 the coefficient is sigma_scat/Vc	0
packing_factor	1	Packing factor (scales the scattering coefficient)	1
interact_fraction	1	Union: fraction of interactions forced to this process (-1: by cross section)	-1
init	string	Deprecated and unused, accepted like on the other Union components (see Union_init).	""

Links

- Component source code found in file `MCViNE_DGSSXRes_process.comp`.
- MCViNE documentation: https://mcvine.github.io

14.145. The `MCViNE_SANS2D_ongrid_process` McStas Component

Union process: SANS from a 2D S(Qx,Qy) map on a grid (beam along z).

Identification

- **Author:** Fahima Islam (McStas port of the MCViNE SANS2D_ongrid_Kernel (mcvine.acc); MCViNE by J. Y. Y. Lin et al.)
- **Origin:** MCViNE, https://github.com/mcvine/acc (acc/kernels/SANS2D_ongrid.py)
- **Date:** 2026-09-29

Description

Qx, Qy are sampled uniformly over the grid range; the neutron gets v_perp -= Q*K2V and vz = sqrt(vi^2 - vx^2 - vy^2) (elastic). The weight is the bilinearly interpolated S(Qx,Qy) (no solid-angle Jacobian, as in mcvine.acc; use scattering_coefficient to set the absolute scale). File layout: first numeric row Qx values (nx, ascending), second row Qy values (ny), then ny rows of nx values (row index = Qy), i.e. the (y,x) image layout used by mcvine.acc.

Union process. Part of the Union components: define this process, collect it into a material with Union_make_material (absorption is set there with my_absorption, the absorption inverse penetration depth at 2200 m/s), assign the material to Union_box/Union_cylinder/Union_sphe geometries, and add a Union_master after them. Geometry, attenuation and multiple scattering are handled by Union_master; this component provides the MCViNE kernel: the scattering coefficient and the final-state sampling (weight). Orientation: this process is anisotropic; its frame (Q vectors, reciprocal vectors, atom positions) follows the ROTATED placement of this component. The kernel code is shared with the standalone component MCViNE_SANS2D_ongrid (mcvine-lib.c). Uses share/mcvine-lib.h/.c and share/mcvine-union-lib.h/.c; the process type MCViNE is declared in share/union-lib.h and registered with Union_master in share/mcvine-union-lib.h.

Example: MCViNE_SANS2D_ongrid_process(S_QxQy="MCViNE/SANS2D_example.dat", scattering_coefficient=1)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
S_QxQy	str	2D grid file (see description)	0
scattering_coefficient	m^{-1}	Scattering coefficient (inverse penetration depth for scattering)	0
sigma_scat	barn	Alternative: scattering cross section per unit cell (used when Vc>0)	0
Vc	$Å^3$	Unit cell volume; when >0 the coefficient is sigma_scat/Vc	0
packing_factor	1	Packing factor (scales the scattering coefficient)	1
interact_fraction	1	Union: fraction of interactions forced to this process (-1: by cross section)	-1
init	string	Deprecated and unused, accepted like on the other Union components (see Union_init).	""

Links

- Component source code found in file `MCViNE_SANS2D_ongrid_process.comp`.
- MCViNE documentation: https://mcvine.github.io

15. Obsolete components

Components in this chapter have been superseded, typically by a more general or more robust replacement, and are kept solely so that existing instrument files referencing them continue to compile and run. **Do not use these in new instrument development.** Where a direct replacement exists it is noted below; run `mcdoc` on the component name for the authoritative, current pointer.

- `V_sample` – superseded by the more general `Incoherent` sample component (see the Samples chapter).
- `Virtual_input`, `Virtual_output`, `Virtual_mcnp_input`, `Virtual_mcnp_output`, `Virtual_mcnp_ss_input`, `Virtual_mcnp_ss_output`, `Virtual_mcnp_ss_Guide`, `Virtual_tripo` `Virtual_tripoli4_output`, `Vitess_input`, `Vitess_output` – code-specific neutron event file readers/writers, superseded by `MCPL_input`/`MCPL_output` (see the Misc chapter), which use the portable, multi-code MCPL event format instead of a McStas/MCNP/Tripoli4/Vitess-specific one. See the User Manual, section on choosing an output data file format.
- `ESS_moderator_short` – an early, simplified ESS moderator model, superseded by `ESS_butterfly` (see the Sources chapter).
- `ESS_moderator` – the 2014–2015 ESS moderator model, superseded by `ESS_butterfly` (see the Sources chapter).
- `Beam_spy` – a diagnostic/debugging component superseded by more general monitor components (e.g. `Monitor_nD`).
- `SasView_model` – the generic interface to SasView's scattering kernels, superseded by the individual `SasView_*` components of the `sasmodels` category (chapter 13).
- `Guide_anyshape_r` – merged into `Guide_anyshape` (see the Optics chapter): per-face m, alpha, W values given in the OFF/PLY file are used with `Guide_anyshape(file_coating`
- `MCPL_input_once` – merged into `MCPL_input` (see the Misc chapter), which by default shares the MCPL file between MPI processes the same way and also supports `repeat_count`. `MCPL_input_once` is kept as an `INHERIT` of `MCPL_input` with `repeat_count=1` forced.

15.1. The `V_sample` McStas Component

Vanadium sample.

Identification

- **Author:** Kim Lefmann and Kristian Nielsen
- **Origin:** Risoe
- **Date:** 15.4.98

Description

A Double-cylinder shaped incoherent scatterer (a V-sample) with both elastic and quasielastic (Lorentzian) components. No multiple scattering. Absorbtion included. The shape of the sample may be a box with dimensions xwidth, yheight, zthick. The area to scatter to is a disk of radius 'focus_r' situated at the target. This target area may also be rectangular if specified focus_xw and focus_yh or focus_aw and focus_ah, respectively in meters and degrees. The target itself is either situated according to given coordinates (x,y,z), or defined with the relative target_index of the component to focus to (next is +1). This target position will be set to its AT position. When targeting to centered components, such as spheres or cylinders, define an Arm component where to focus to.

Example: V_sample(radius_i=0.001,radius_o=0.01,h=0.02,focus_r=0.035,pack=1, target_index=1)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
radius	m	Outer radius of sample in (x,z) plane	0
thickness	m	Thickness of outer wall	0
zdepth	m	depth of box sample	0
Vc	$Å^3$	Unit cell volume	13.827
sigma_abs	barns	Absorbtion cross section pr. unit cell	5.08
sigma_inc	barns	Incoherent scattering cross section pr. unit cell	5.08
radius_i	m	radius-thickness	0
radius_o	m	Same as radius	0
h	m	Same as yheight	0
focus_r	m	Radius of disk containing target. Use 0 for full space	0
pack	1	Packing factor	1
frac	1	MC Probability for scattering the ray; otherwise penetrate	1
f_QE	1	Fraction of quasielastic scattering (rest is elastic)	0

Name	Unit	Description	Default
gamma	1	Lorentzian width of quasielastic broadening (HWHM)	0
target_x	m	position of target to focus at	0
target_y	m	position of target to focus at	0
target_z	m	position of target to focus at	0
focus_xw	m	horiz. dimension of a rectangular area	0
focus_yh	m	vert. dimension of a rectangular area	0
focus_aw	deg	angular width dimension of a rectangular area	0
focus_ah	deg	angular height dimension of a rectangular area	0
xwidth	m	horiz. dimension of sample	0
yheight	m	vert. dimension of sample	0
zthick	m	Same as zdepth	0
rad_sphere	m	Radius for a spherical sample	0
sig_a	barns	Same as sigma_abs	0
sig_i	barns	Same as sigma_inc	0
V0	$Å^3$	Same as Vc	0
target_index	1	relative index of component to focus at, e.g. next is +1	0
multiples	1	Apply crude estimate for multiple scattering	1

Links

- Component source code found in file `V_sample.comp`.
- <A HREF="http://neutron.risoe.dk/mcstas/components/tests/v_sample/">Test
- results</A> (not up-to-date).
- The test/example instrument vanadium_example.instr.
- The test/example instrument QENS_test.instr.

15.2. The `Virtual_input` McStas Component

Source-like component that generates neutron events from an ascii 'virtual source' filename.

Identification

- **Author:** E. Farhi

- **Origin:** ILL
- **Date:** Sep 28th, 2001

Description

This component reads neutron events stored from a file, and sends them into the instrument. It thus replaces a Source component, using a previously computed neutron set. The 'source' file type is an ascii text file with the format listed below. The number of neutron events for the simulation is set to the length of the 'source' file times the repetition parameter 'repeat_count' (1 by default). It is particularly useful to generate a virtual source at a point that few neutron reach. A long simulation will then only be performed once, to create the 'source' filename. Further simulations are much faster if they start from this low flux position with the 'source' filename.

The input file format is: text column formatted with lines containing 11 values in the order: p x y z vx vy vz t sx sy sz stored into about 83 bytes/n.

%BUGS We recommend NOT to use parallel execution (MPI) with this component. If you need to, set parameter 'smooth=1'.

EXAMPLE: To create a 'source' file collecting all neutron states, use: COMPONENT MySourceCreator = Virtual_output(filename = "MySource.list") at the position where will be the Virtual_input. Then inactivate the part of the simulation description before (and including) the component MySourceCreator. Put the new instrument source: COMPONENT Source = Virtual_input(filename="MySource.list") at the same position as 'MySourceCreator'. A Vitess filename may be obtained from the 'Vitess_output' component or from a Vitess simulation (104 bytes per neutron) and read with Vitess_input.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename	str	Name of the neutron input file. Empty string "" inactivates component	0
verbose	0/1	Display additional information about source, recommended	0
repeat_count	1	Number of times the source must be generated/repeated	1
smooth	0/1	Smooth sparsed event files for filename repetitions. Use this option with MPI. This will apply gaussian distributions around initial events from the file	1

Links

- Component source code found in file `Virtual_input.comp`.

15.3. The `Virtual_output` McStas Component

Detector-like component that writes neutron state parameters into an ascci-format 'virtual source' neutron file.

Identification

- **Author:** E. Farhi
- **Origin:** ILL
- **Date:** Dec 17th, 2002

Description

Detector-like component writing neutron state parameters to a virtual source neutron filename. The component geometry is the full plane, and saves the neutron state as it exits from the previous component.

It is particularly useful to generate a virtual source at a point that few neutron reach. A long simulation will then only be performed once, to create the upstream 'source' file. Further simulations are much faster if they start from this low flux position with the 'source' filename.

The output file format is: text column formatted with lines containing 11 values in the order: p x y z vx vy vz t sx sy sz stored into about 83 bytes/n.

Beware the size of generated files ! When saving all events (bufsize=0) the required memory has been optimized and remains very small. On the other hand using large bufsize values (not recommended) requires huge storage memory. Moreover, using the 'bufsize' parameter will often lead to wrong intensities. Both methods will generate huge files.

A Vitess file may be obtained from the 'Vitess_output' component or from a Vitess simulation (104 bytes per neutron) and read with Vitess_input.

Example: Virtual_output(filename="MySource.dat") will generate a 9 Mo text file for 1e5 events stored.

%BUGS Using bufsize non-zero may generate a virtual source with wrong intensity. This component works with MPI (parallel execution mode).

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename	str	Name of neutron file to write. Default is standard output [string]. If not given, a unique name will be used.	0

Name	Unit	Description	Default
bufsize	1	Size of neutron output buffer default is 0, i.e. save all - recommended.	0

Links

- Component source code found in file `Virtual_output.comp`.

15.4. The `Virtual_mcnp_input` McStas Component

This component uses a filename of recorded neutrons from the reactor monte carlo code MCNP as a source of particles.

Identification

- **Author:** Chama Hennane and E. Farhi
- **Origin:** ILL
- **Date:** June 28th, 2006

Description

This component generates neutron events from a filename created using the MCNP Monte Carlo code for nuclear reactors. It is used to calculate flux exiting from hot or cold neutron sources. Neutron position and velocity is set from the filename. The neutron time is left at zero.

Note that axes orientation may be different between MCNP and McStas. The component has the ability to center and orient the neutron beam to the Z-axis. It also may change the coordinate system from the MCNP frame to the McStas one. The verbose mode is highly recommended as it displays lots of useful informations. To obtain absolute intensity, set 'intensity' and 'nps' parameters. The source total intensity is 1.054e18 for LLB/Saclay (14 MW) and 4.28e18 for ILL/Grenoble (58 MW).

Format of MCNP events are :

position_X position_Y position_Z dir_X dir_Y dir_Z Energy Weight Time

energy is in Mega eV, time in shakes (1e-8 s), positions are in cm and the direction vector is normalized to 1.

%BUGS We recommend NOT to use parallel execution (MPI) with this component. If you need to, set parameter 'smooth=1'.

EXAMPLE: To generate PTRAC files using MCNP/MCNPX, add at the end of your input file:

```
f1:n          2001              // tally
```

(...)

```
ptrac          filename = asc
max  = -1000000   // number of neutrons to generate and stop
```

write = all event = sur filter = 2001,jsu // surface tally id To create a 'source' from a MCNP simulation event file for the ILL: COMPONENT source = Virtual_mcnp_input(filename = "H10p", intensity=4.28e18, nps=11328982, verbose = 1, autocenter="translate rotate rescale")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename	str	Name of the MCNP PTRAC neutron input file. Empty string "" unactivates component	0
autocenter	str	String which may contain following keywords. "translate" or "center" to center the beam area AT (0,0,0) "rotate" or "orient" to center the beam velocity along Z "change axes" to change coordinate system definition "rescale" to adapt intensity to abs. units. with factor intensity/nps. Other words are ignored.	0
repeat_count	1	Number of times the source must be generated. 0 unactivates the component	1
verbose	0|1	Displays additional informations if set to 1	0
intensity	n/s	Intensity multiplication factor	0
MCNP_ANALYSE	1	Number of neutron events to read for file pre-analysis. Use 0 to analyze them all.	10000
surface_id	1	Index of the emitting MCNP surface to use. -1 for all.	-1
nps	1	Number of total events shot by MCNP to generate the PTRAC as indicated at the end of the MCNP 'o' file as 'source particle weight for summary table normalization' or alternatively 'nps = '.	0
smooth	0/1	Smooth sparsed event files for file repetitions.	1

Links

- Component source code found in file `Virtual_mcnp_input.comp`.
- MCNP
- MCNP – A General Monte Carlo N-Particle Transport Code, Version 5, Volume II: User's Guide, p177

15.5. The `Virtual_mcnp_output` McStas Component

Detector-like component that writes neutron state parameters into a 'virtual source' neutron file with MCNP/PTRAC format.

Identification

- **Author:** Chama Hennane and E. Farhi
- **Origin:** ILL
- **Date:** July 7th, 2006

Description

Detector-like component writing neutron state parameters to a virtual source neutron file with MCNP/PTRAC format. The component geometry is the full plane, and saves the neutron state as it exits from the previous component. Format is the one used by MCNP 5 files :

position_X position_Y position_Z dir_X dir_Y dir_Z energy weight time

energy is in Mega eV positions are in cm and the direction vector is normalized to 1.

%BUGS This component will NOT work with parallel execution (MPI).

EXAMPLE: To create a file collecting all neutron states with MCNP5 format COMPONENT fichier_sortie = Virtual_mcnp_output(filename = "exit_guide_result.dat") at the position where will be the Virtual_mcnp_input.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename	str	name of the MCNP5 neutron output file, or stdout if left to 0.	0

Links

- Component source code found in file `Virtual_mcnp_output.comp`.
- MCNP
- MCNP – A General Monte Carlo N-Particle Transport Code, Version 5, Volume II: User's Guide, p177

15.6. The `Virtual_mcnp_ss_input` McStas Component

This component uses a Source Surface type file of recorded neutrons from the reactor monte carlo code MCNP as a source of particles.

Identification

- **Author:** Esben Klinkby and <a href="mailto:pkwi@dtu
  Willendrup
- **Origin:** DTU
- **Date:** January, 2012

Description

This component draws neutron events from a Source Surface file created using the MCNP Monte Carlo code and converts them to make them suitable for a McStas simulation

Note that axes orientation may be different between MCNP and McStas! Note also that the conversion of between McStas and MCNP units and parameters is done automatically by this component - but the user must ensure that geometry description matches between the two Monte Carlo codes.

The verbose mode is highly recommended as it displays lots of useful informations.

This interface uses the MCNP Source Surface Read/Write format (SSW/SSR). Infomation transfer from(to) SSW files proceeds via a set of Fortran modules and subroutines collected in "subs.f" For succesful compilation, it is required that these subroutines are compiled and linked to the instrument file:

mcstas dummy.instr -> generates dummy.c gfortran -c subs.f -> generates subs.f gcc -o runme.out dummy.c subs.o -lm -lgfortran -> generates runme.out

Note that this requires a fortran compiler (here gfortran) and gcc.

%BUGS None known bugs so far. But surely this will change... Caveat: when writing the header for the output wssa file, the number of histories and tracks are assumed to be that of the input MNCP run. This is generally not the case, due to losses in the McStas simulation step. In case of losses any subsequel MCNP run based on the McStas output, can be confused by inconsistency between header and file content. To resolve, either ensure that NPS is lower than the actual number of events in the McStas output. Or hardcode values of nhis & ntrk in either subs.f or Virtual_mcnp_ss_output.comp.

EXAMPLE of usage: first a MCNP simulation is run and at a relevant surface, a Source Surface Write card is given (e.g. for MCNP surface #1, add the line: SSW 1 to the end of the input file. By this a ".w" file is produced, which then serves as input to the McStas simulation. Present version: rename *.w to rssa, and make sure to put it in the dir from which McStas is run

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
file	str	Name of the MCNP SSW neutron input file. Not active: Name assumed to be 'rssa'	"rssa"
verbose	1	Toggles debugging info on/off	1

Links

- Component source code found in file `Virtual_mcnp_ss_input.comp`.
- MCNP
- MCNP – A General Monte Carlo N-Particle Transport Code, Version 5, Volume II: User's Guide, p177

15.7. The `Virtual_mcnp_ss_output` McStas Component

This component uses a Source Surface type file of recorded neutrons from the reactor monte carlo code MCNP as a source of particles.

Identification

- **Author:** Esben Klinkby and Peter Willendrup
- **Origin:** DTU
- **Date:** January, 2012

Description

This component draws neutron events from a Source Surface file created using the MCNP Monte Carlo code and converts them to make them suitable for a McStas simulation

Note that axes orientation may be different between MCNP and McStas! Note also that the conversion of between McStas and MCNP units and parameters is done automatically

by this component - but the user must ensure that geometry description matches between the two Monte Carlo codes.

The verbose mode is highly recommended as it displays lots of useful informations.

This interface uses the MCNP Source Surface Read/Write format (SSW/SSR). Infomation transfer from(to) SSW files proceeds via a set of Fortran modules and subroutines collected in "subs.f" For succesful compilation, it is required that these subroutines are compiled and linked to the instrument file:

mcstas dummy.instr -> generates dummy.c gfortran -c subs.f -> generates subs.f gcc -o runme.out dummy.c subs.o -lm -lgfortran -> generates runme.out

Note that this requires a fortran compiler (here gfortran) and gcc.

%BUGS None known bugs so far. But surely this will change... Caveat: when writing the header for the output wssa file, the number of histories and tracks are assumed to be that of the input MNCP run. This is generally not the case, due to losses in the McStas simulation step. In case of losses any subsequel MCNP run based on the McStas output, can be confused by inconsistency between header and file content. To resolve, either ensure that NPS is lower than the actual number of events in the McStas output. Or hardcode values of nhis & ntrk in either subs.f or Virtual_mcnp_ss_output.comp.

EXAMPLE of usage: first a MCNP simulation is run and at a relevant surface, a Source Surface Write card is given (e.g. for MCNP surface #1, add the line: SSW 1 to the end of the input file. By this a ".w" file is produced, which then serves as input to the McStas simulation. Present version: rename *.w to rssa, and make sure to put it in the dir from which McStas is run (or use symbolic links)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
file	str	Name of the MCNP SSW neutron input file. Not active: Name assumed to be 'rssa'	"rssa"
verbose	1	Toggles debugging info on/off	1

Links

- Component source code found in file `Virtual_mcnp_ss_output.comp`.
- MCNP
- MCNP – A General Monte Carlo N-Particle Transport Code, Version 5, Volume II: User's Guide, p177

15.8. The `Virtual_mcnp_ss_Guide` McStas Component

Neutron guide initiated using Virtual_mcnp_ss_input.comp, and replacing Virtual_mcnp_ss_output.comp - see examples//Test_SSR_SSW_Guide.instr

Identification

- **Author:** Esben klinkby and Peter Willendrup
- **Origin:** Risoe-DTU
- **Date:** Marts 2012

Description

Based on Kristian Nielsens Guide.comp Models a rectangular guide tube centered on the Z axis. The entrance lies in the X-Y plane. The component must be initiated after Virtual_mcnp_ss_input.comp, and replaces Virtual_mcnp_ss_output.comp - see examples/Test_SSR_SSW_Guide.instr. The basic idea is, that rather than discarding unreflected (i.e. absorbed) neutrons at the guide mirrors, these neutron states are stored on disk. Thus, after the McStas simulation a MCNP simulation can be performed based on the un-reflected neutrons - intended for shielding studies. (details: we don't deal with actual neutrons, so what is transferred between simulations suites is neutron state parameters: pos,mom,time,weight. The latter is whatever remains after reflection.

For details on the geometry calculation see the description in the McStas reference manual. The reflectivity profile may either use an analytical mode (see Component Manual) or a 2-columns reflectivity free text file with format

```
[q(Angs-1) R(0-1)].
```

Example: Virtual_mcnp_ss_Guide(w1=0.1, h1=0.1, w2=0.1, h2=0.1, l=2.0,

```
R0=0.99, Qc=0.021, alpha=6.07, m=2, W=0.003
```

%VALIDATION Upcomming in 2012 based on ESS shielding Validated by: D. Ene & E. Klinkby

%BUGS This component does not work with gravitation on. Use component Guide_gravity then (doesn't work with SSR/SSW unfortunately)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
reflect	str	Reflectivity file name. Format <q(Å-1) R(0-1)>	0
w1	m	Width at the guide entry	
h1	m	Height at the guide entry	
w2	m	Width at the guide exit	
h2	m	Height at the guide exit	
l	m	length of guide	
R0	1	Low-angle reflectivity	0.99
Qc	$Å^{-1}$	Critical scattering vector	0.0219
alpha	Å	Slope of reflectivity	6.07
m	1	m-value of material. Zero means completely absorbing.	2
W	$Å^{-1}$	Width of supermirror cut-off	0.003

Links

- Component source code found in file `Virtual_mcnp_ss_Guide.comp`.

15.9. The `Virtual_tripoli4_input` McStas Component

This component reads a file of recorded neutrons from the reactor Monte Carlo code TRIPOLI4.4 as a source of particles.

Identification

- **Author:** Guillaume Campioni
- **Origin:** SERMA
- **Date:** Sep 28th, 2001

Description

This component generates neutron events from a file created using the TRIPOLI4 Monte Carlo code for nuclear reactors (as MCNP). It is used to calculate flux exiting from hot or cold neutron sources. Neutron position and velocity is set from the file. The neutron time is left at zero. Storage files from TRIPOLI4.4 contain several batches of particules, all of them having the same statictical weight.

Note that axes orientation may be different between TRIPOLI4.4 and McStas. The component has the ability to center and orient the neutron beam to the Z-axis. It also changes the coordinate system from the Tripoli frame to the McStas one. The verbose mode is highly recommended as it displays lots of useful informations, including the absolute intensity normalisation factor. All neutron fluxes in the instrument should be

multiplied by this factor. Such a renormalization is done when 'autocenter' contains the word 'rescale'. The source total intensity is 1.054e18 for LLB/Saclay (14 MW) and 4.28e18 for ILL/Grenoble (58 MW).

Format of TRIPOLI4.4 event files is :

NEUTRON energy position_X position_Y position_Z dir_X dir_Y dir_Z weight

energy is in Mega eV positions are in cm and the direction vector is normalized to 1.

%BUGS We recommend NOT to use parallel execution (MPI) with this component. If you need to, set parameter 'smooth=1'.

EXAMPLE: To create a 'source' from a Tripoli4 simulation event file for the ILL: COMPONENT source = Virtual_tripoli4_input(filename = "ILL_SFH.dat", intensity=4.28e18, verbose = 1, autocenter="translate rotate rescale")

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename	str	Name of the Tripoli4.4 neutron input file. Empty string "" unactivates component	0
autocenter	str	String which may contain following keywords. "translate" or "center" to center the beam area AT (0,0,0) "rotate" or "orient" to center the beam velocity along Z "rescale" to adapt intensity to abs. units. Other words are ignored.	0
repeat_count	1	Number of times the source must be generated. 0 unactivates the component	1
verbose	0|1	Displays additional informations if set to 1	0
intensity	n/s	Initial total Source integrated intensity	1
T4_ANALYSE	1	Number of neutron events to read for file pre-analysis. Use 0 to analyze them all.	10000
radius	m	In the case the Tripoli4 batch file is a point, you may specify a disk emission area for the source.	0
T4_ANALYSE_EMIN	MeV	Minimal energy to use for file pre-analysis	0
T4_ANALYSE_EMAX	MeV	Maximal energy to use for file pre-analysis	0
smooth	0/1	Smooth sparsed event files for file repetitions.	1

Links

- Component source code found in file `Virtual_tripoli4_input.comp`.
- Tripoli
- Virtual_tripoli4_output
- CAMPIONI Guillaume, Etude et Modelisation des Sources Froides de Neutrons, These de Doctorat, CEA Saclay/UJF (2004)

15.10. The `Virtual_tripoli4_output` McStas Component

Detector-like component that writes neutron state parameters into a 'virtual source' neutron file when neutrons come from the source : Virtual_tripoli4_input.comp

Identification

- **Author:** Guillaume Campioni
- **Origin:** LLB
- **Date:** Sep 28th, 2001

Description

Detector-like component writing neutron state parameters to a virtual source neutron file when neutron are coming from a Virtual_tripoli4_input.comp. The component geometry is the full plane, and saves the neutron state as it exits from the previous component. Format is the one used by TRIPOLI4.4 stock files :

NEUTRON energy position_X position_Y position_Z dir_X dir_Y dir_Z weight

energy is in [Mega eV] positions are in [cm] and the direction vector is normalized to 1.

%BUGS This component will NOT work with parallel execution (MPI/GPU).

EXAMPLE: To create a file collecting all neutron states with TRIPOLI4 format COMPONENT T4output = Virtual_tripoli4_output(filename = "exit_guide_result.dat", batch = 1) at the position where will be the Virtual_tripoli4_input when reading the file.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename	str	Name of the Tripoli4 neutron output file, or stdout if left to 0.	0

Name	Unit	Description	Default
batch	1	Index of the Tripoli batch to generate, when no Tripoli4_input component is available in instrument.	1

Links

- Component source code found in file `Virtual_tripoli4_output.comp`.
- Tripoli
- Virtual_tripoli4_input

15.11. The `Vitess_input` McStas Component

Read neutron state parameters from VITESS neutron filename.

Identification

- **Author:** Kristian Nielsen
- **Origin:** Risoe/ILL
- **Date:** June 6, 2000

Description

Source-like component reading neutron state parameters from a VITESS neutron filename. Used to interface McStas components or simulations into VITESS. Each neutron is 104 bytes.

Example: Vitess_input(filename="MySource.vit", bufsize = 10000, repeat_count = 2)

%BUGS We recommend NOT to use parallel execution (MPI) with this component.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename	string	Filename of neutron file to read. Default (NULL) is standard input. Empty string "" unactivates component	0
bufsize	records	Size of neutron input buffer	10000

Name	Unit	Description	Default
repeat_count	1	Number of times to repeat each neutron read	1

Links

- Component source code found in file `Vitess_input.comp`.

15.12. The `Vitess_output` McStas Component

Write neutron state parameters to VITESS neutron filename.

Identification

- **Author:** Kristian Nielsen
- **Origin:** Risoe/ILL
- **Date:** June 6, 2000

Description

Detector-like component writing neutron state parameters to a VITESS neutron filename. Used to interface McStas components or simulations into VITESS. Each neutron is 104 bytes.

Note that when standard output is used, as is the default, no monitors or other components that produce terminal output must be used, or the neutron output from this component will become corrupted.

Example: Vitess_output(filename="MySource.vit", bufsize = 10000, progress = 1)

%BUGS This component will NOT work with parallel execution (MPI).

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
filename	string	Filename of neutron file to write. Default is standard output	0
bufsize	records	Size of neutron output buffer	10000
progress	flag	If not zero, output dots as progress indicator	0

Links

- Component source code found in file `Vitess_output.comp`.

15.13. The `ESS_moderator_short` McStas Component

A parametrised pulsed source for modelling ESS short pulses.

Identification

- **Author:** KL, February 2001
- **Origin:** Risoe
- **Date:**

Description

Produces a time-of-flight spectrum, from the ESS parameters Chooses evenly in lambda, exponentially decaying in time . Adapted from Moderator by: KN, M.Hagen, August 1998

Units of flux: n/cm^2/s/Å/ster (McStas units are in general neutrons/second)

Example general parameters (general): size=0.12 Lmin=0.1 Lmax=10 dist=2 focus_xw=0.06 yh=0.12 nu=50 branchframe=0.5

Example moderator specific parameters (From F. Mezei, "ESS reference moderator characteristics for ...", 4/12/00: Defining the normalised Maxwelian M(lam,T) = 2 a^2 lam^-5 exp(-a/lam^2); a=949/T; lam in Å; T in K, and the normalised pulse shape function F(t,tau,n) = (exp(-t/tau) - exp(-nt/tau)) n/(n-1)/tau, the flux distribution is given as Phi(t,lam) = I0 M(lam,T) F(t,tau,n) + I2/(1+exp(chi2 lam-2.2))/lam*F(t,tau2*lam,n2))

a1: Ambient H20, short pulse, decoupled poisoned

```
T=325 tau=22e-6 tau1=0 tau2=7e-6 n=5 n2=5 chi2=2.5
I0=9e10 I2=4.6e10    branch1=0 branch2=0.5
```

a2: Ambient H20, short pulse, decoupled un-poisoned

```
T=325 tau=35e-6 tau1=0 tau2=12e-6 n=5 n2=5 chi2=2.5
I0=1.8e11 I2=9.2e10    branch1=0 branch2=0.5
```

a3: Ambient H20, short pulse, coupled

```
T=325 tau=80e-6 tau1=400e-6 tau2=12e-6 n=20 n2=5 chi2=2.5
I0=4.5e11 I2=9.2e10    branch1=0.5 branch2=0.5
```

b1: Liquid H2, short pulse, decoupled poisoned

```
T=50 tau=49e-6 tau1=0 tau2=7e-6 n=5 n2=5 chi2=0.9
I0=2.7e10 I2=4.6e10    branch1=0 branch2=0.5
```

b2: Liquid H2, short pulse, decoupled un-poisoned

```
T=50 tau=78e-6 tau1=0 tau2=12e-6 n=5 n2=5 chi2=0.9
I0=5.4e10 I2=9.2e10    branch1=0 branch2=0.5
```

b3: Liquid H2, short pulse, coupled

```
T=50 tau=287e-6 tau1=0 tau2=12e-6 n=20 n2=5 chi2=0.9
I0=2.3e11 I2=9.2e10    branch1=0 branch2=0.5
```

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
size	m	Edge of cube shaped source	
Lmin	Å	Lower edge of wavelength distribution	
Lmax	Å	Upper edge of wavelength distribution	
dist	m	Distance from source to focusing rectangle; at (0,0,dist)	
focus_xw	m	Width of focusing rectangle	
focus_yh	m	Height of focusing rectangle	
nu	Hz	Frequency of pulses	50
T	K	Temperature of source	325
tau	s	long time decay constant for pulse, 1a	22e-6
tau1	s	long time decay constant for pulse, 1b (only for coupled water, else 0)	0
tau2	s/Å	long time decay constant for pulse, 2	7e-6
n	1	pulse shape parameter 1	5
n2	1	pulse shape parameter 2	5
chi2	1/Å	lambda-distribution parameter in pulse 2	2.5
I0	flux	integrated flux 1 (in flux units, see above) (default 9e10)	9e10
I2	flux*Å	integrated flux 2 (default 4.6e10)	4.6e10
branch1	1	limit for switching between distribution 1 and 2. (has effect only for coupled water, tau1>0)	0
branch2	1	limit for switching between distribution 1 and 2. (default value 0.5)	0.5
branchframe	1	limit for switching between 1st and 2nd pulse (if only one pulse wanted: 1)	0

Name	Unit	Description	Default
target_index	1	relative index of component to focus at, e.g. next is +1 this is used to compute 'dist' automatically.	1

Links

- Component source code found in file `ESS_moderator_short.comp`.

15.14. The `ESS_moderator` McStas Component

A parametrised pulsed source for modelling ESS long pulses.

Identification

- **Author:** P Willendrup and E Klinkby, February 2014, derived from K Lefmann ESS_moderator_long
- **Origin:** DTU
- **Date:**

Description

Produces a time-of-flight spectrum, from the ESS parameters Chooses evenly in lambda, evenly/exponentially decaying in time Adapted from ESS_moderator_long by: K Lefmann, 2001

Updates and simplified interface: <ol> The spectrum from the source(s) is defined via the sourcedef string input parameter which allows these values: <ul> sourcedef="2001", legacy "Mezei moderators" from the original F. Mezei documents "ESS reference moderator characteristics for generic instrument performance evaluation", but rescaled to ESS TDR frequency, pulselength and power. sourcedef="TDR", Mezei moderators, with a wavelength-dependent correction term to the cold flux, derived from 2012 MCNPX calculations by ESS neutronics group. Corrections calculated by K Lieutenant (Vitess) and implemented here by E Klinkby. NOTE: uses the 2001 brilliance for the thermal moderator! sourcedef="2014", updated brilliance using formulation by Troels Schoenfeldt, including support for the "pancacke", i.e. flat geometry. sourcedef="2015", updated brilliance using formulation by Troels Schoenfeldt, new butterfly baseline. The component can use target_index for focusing to a given beam port. Use an Arm() and ROTATED to position relatively to the moderator. The component relies on the new ess_source-lib which is expected to become further enriched during design-finaliziation and construciton of the ESS.

<p>**Note that this component does not implement "engineering reality" and currently uses a coordinate system centered on the moderator assembly.**

An updated moderator component which references the "Moderator focus coordinate system" will be released later during the spring of 2016.

<p>Derived from ESS_moderator_long which was debugged intensively against Mezei note (4/12 2000) and VitESS @ Rencurel 2006.

Correction by J. Saroun, NPI Rez: 1) version 2015: accepts negative port angles 2) version 2015: weight by cosine of the port angle Warning: The negative beamport angle is not taken into acccount by mcplot

%VALIDATION Mezei-modererators validated against VitESS and Mezei note (4/12 2000) @ Rencurel 2006 Benchmarked against multiple versions of ESS moderator group simulation data 2013-2015

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
isleft	1	Fraction of thermal neutrons generated at the "left" moderator slab in case of "2013" or "2014"	0.9
Lmin	Å	Lower edge of wavelength distribution	
Lmax	Å	Upper edge of wavelength distribution	
cold_frac	1	Fraction of neutron statistics from cold source. It is implicitely assumed that supermirror allows each beamline to choose the desired fraction of cold and thermal neutrons (i.e. extreme idealization).	1.0
dist	m	Distance from source to focusing rectangle; at (0,0,dist)	0
focus_xw	m	Width of focusing rectangle	
focus_yh	m	Height of focusing rectangle	
target_index	1	relative index of component to focus at, e.g. next is +1 this is used to compute 'dist' automatically.	0
tmax_multiplier	1	Defined maximum emission time at moderator, tmax= tmax_multiplier * ESS_PULSE_DURATION. Only in combination with sourcedef="2013", "2014" or "2015"	3
yheight_c	m	Height of the cold source	0.03
yheight_t	m	Height of the thermal source	0.03

Name	Unit	Description	Default
n_pulses	1	Number of pulses simulated. 0 and 1 creates one pulse. The integrated intensity is constant	1
acc_power	MW	Accelerator power in MW	5
beamport_angle	deg	Direction within the beamport sector (0 < angle < extraction_opening for 2014, -extraction_opening/2 < angle < extraction_opening/2 for 2015) to direct neutrons. For sourcedef="2015", the only allowed values are 5,15,...,55 degrees measured from the central point.	25
sourcedef	string	ESS source "database", values: "TDR", "2001", "2013", "2014", "2015"	"2015"
xwidth_c	m	Width / arc-length opening of the cold source.	0.1
xwidth_t	m	Edge of thermal source	0.18
extraction_opening	deg	Width of extraction-area in degrees (60 or 120 degrees). 120 deg only in combination with sourcedef="2014" and "2015".	120

Links

- Component source code found in file `ESS_moderator.comp`.

15.15. The `Beam_spy` McStas Component

Beam analyzer for previous component

Identification

- **Author:** E. Farhi
- **Origin:** Risoe
- **Date:** Nov 2005

Description

This component displays informations about the beam at the previous component position. No data file is produced. It behaves as the Monitor component, but No propagation to the Beam_spy is performed, and all events are analyzed. The component should be located at the same position as the previous one.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default

Links

- Component source code found in file `Beam_spy.comp`.
- Monitor component

15.16. The `SasView_model` McStas Component

This SANS sample exposes SasView's scattering kernels to McStas. In this way SasView's monodisperse scattering kernels can be call from McStas.

Identification

- **Author:** Jakob Garde, Torben Nielsen, Peter Willendrup
- **Origin:** SasView, DTU, European Spallation Source ERIC
- **Date:** 03.02.2016

Description

OBSOLETE - we recommend using individual component models form the <pre>sasmodels</pre> selection. If you keep using this component, be aware that the URLs listed in the table below are broken, please refer to the new revamped SasView model page.

Sample for use in SANS instruments. The models describe mono disperse particles in thin solution. The sample geometry may have the shape:

```
Shape:        - A filled box with dimensions xwidth, yheight and zdepth.
```

- A cylinder with dimensions radius and yheight. - A filled sphere given by radius.

These parameters are mutually exclusive.

Example using spheres in thin solution, with radius=200 Å and a delta_sld=0.6 fm/Å^3: SasView_model(model_index=47, model_scale=1.0, model_pars={1, 7, 200}, model_abs=0.0, xwidth=0.01, yheight=0.01, zdepth=0.005,)

The algorithm of this component requires use of the ISO C standard c99, standard from McStas 2.7 and 3.0.

The list of scattering models in SasView is called sasmodels. The list of McStas available SasView sasmodels are found in the table below.

A few models may require manual documentation lookup using the above link to the SasView site.

McStas does currently not support multiplication of formfactor models with structure factor models.

The 2D scattering scattering kernels are denoted by modelname_xy. I.e. to evalulate scattering from aligned cylinders use model_index=10 to use cylinder_xy.

MDOC

<table border=1><tr><td>**Model no.**</td><td>**SasView name**</td><td>**Parameters**</td></tr>
<tr><td> 0</td><td>None</td><td>None</td></tr> <tr><td> 1</td><td>barbell</td><td>(sld,
solvent_sld, bell_radius, radius, length)</td></tr> <tr><td> 2</td><td>barbell_xy</td><td>(sld,
solvent_sld, bell_radius, radius, length, theta, phi)</td></tr> <tr><td> 3</td><td>bcc_paracrystal</td>
d_factor, radius, sld, solvent_sld)</td></tr> <tr><td> 4</td><td>bcc_paracrystal_xy</td><td>(dnn,
d_factor, radius, sld, solvent_sld, theta, phi, psi)</td></tr> <tr><td> 5</td><td>capped_cylinder</td><
solvent_sld, radius, cap_radius, length)</td></tr> <tr><td> 6</td><td>capped_cylinder_xy</td><td>(
solvent_sld, radius, cap_radius, length, theta, phi)</td></tr> <tr><td> 7</td><td>core_shell_cylinder</t
shell_sld, solvent_sld, radius, thickness, length)</td></tr> <tr><td> 8</td><td>core_shell_cylinder_xy<
shell_sld, solvent_sld, radius, thickness, length, theta, phi)</td></tr> <tr><td>
9</td><td>cylinder</td><td>(sld, solvent_sld, radius, length)</td></tr> <tr><td>10</td><td>cylinde
solvent_sld, radius, length, theta, phi)</td></tr> <tr><td>11</td><td>dab</td><td>(length)</td></t
<tr><td>12</td><td>dab_xy</td><td>(length)</td></tr> <tr><td>13</td><td>ellipsoid</td><td
solvent_sld, rpolar, requatorial)</td></tr> <tr><td>14</td><td>ellipsoid_xy</td><td>(sld,
solvent_sld, rpolar, requatorial, theta, phi)</td></tr> <tr><td>15</td><td>fcc_paracrystal</td><td>(d
d_factor, radius, sld, solvent_sld)</td></tr> <tr><td>16</td><td>fcc_paracrystal_xy</td><td>(dnn,
d_factor, radius, sld, solvent_sld, theta, phi, psi)</td></tr> <tr><td>17</td><td>flexible_cylinder_ex</t
kuhn_length, radius, axis_ratio, sld, solvent_sld)</td></tr> <tr><td>18</td><td>flexible_cylinder_ex_x
kuhn_length, radius, axis_ratio, sld, solvent_sld)</td></tr> <tr><td>19</td><td>gaussian_peak</td><
sigma)</td></tr> <tr><td>20</td><td>gaussian_peak_xy</td><td>(q0, sigma)</td></tr>
<tr><td>21</td><td>guinier</td><td>(rg)</td></tr> <tr><td>22</td><td>guinier_xy</td><td>(
<tr><td>23</td><td>hardsphere</td><td>(effect_radius, volfraction)</td></tr>
<tr><td>24</td><td>hardsphere_xy</td><td>(effect_radius, volfraction)</td></tr>
<tr><td>25</td><td>HayterMSAsq</td><td>(effect_radius, zz, VolFrac, Temp,
csalt, dialec)</td></tr> <tr><td>26</td><td>HayterMSAsq_xy</td><td>(effect_radius,
charge, volfraction, temperature, saltconc, dielectconst)</td></tr> <tr><td>27</td><td>hollow_cylinder</
core_radius, length, sld, solvent_sld)</td></tr> <tr><td>28</td><td>hollow_cylinder_xy</td><td>(ra
core_radius, length, sld, solvent_sld, theta, phi)</td></tr> <tr><td>29</td><td>lamellar</td><td>(sld,
solvent_sld, thickness)</td></tr> <tr><td>30</td><td>lamellar_xy</td><td>(sld,
solvent_sld, thickness)</td></tr> <tr><td>31</td><td>lamellar_FFHG</td><td>(tail_length,
head_length, sld, head_sld, solvent_sld)</td></tr> <tr><td>32</td><td>lamellar_FFHG_xy</td><td>
head_length, sld, head_sld, solvent_sld)</td></tr> <tr><td>33</td><td>lamellarCailleHG</td><td>(ta
head_length, Nlayers, dd, Cp, tail_sld, head_sld, solvent_sld)</td></tr> <tr><td>34</td><td>lamellarCa
head_length, Nlayers, spacing, Caille_parameter, sld, head_sld, solvent_sld)</td></tr>
<tr><td>35</td><td>lamellarPC</td><td>(th, Nlayers, davg, pd, sld, solvent_sld)</td></tr>
<tr><td>36</td><td>lamellarPC_xy</td><td>(thickness, Nlayers, spacing, spac-
ing_polydisp, sld, solvent_sld)</td></tr> <tr><td>37</td><td>lamellarPS</td><td>(del,

Nlayers, dd, Cp, sld, solvent_sld)</td></tr> <tr><td>38</td><td>lamellarPS_xy</td><td>
Nlayers, spacing, Caille_parameter, sld, solvent_sld)</td></tr> <tr><td>39</td><td>linear_p
edge_sep, num_pearls, pearl_sld, solvent_sld)</td></tr> <tr><td>40</td><td>linear_pearls
edge_sep, num_pearls, pearl_sld, solvent_sld)</td></tr> <tr><td>41</td><td>lorentz</td>
<tr><td>42</td><td>lorentz_xy</td><td>(cor_length)</td></tr> <tr><td>43</td><td
mass_dim, cutoff_length)</td></tr> <tr><td>44</td><td>mass_fractal_xy</td><td>(radi
mass_dim, cutoff_length)</td></tr> <tr><td>45</td><td>mass_surface_fractal</td><td>
surface_dim, cluster_rg, primary_rg)</td></tr> <tr><td>46</td><td>mass_surface_fractal_
surface_dim, cluster_rg, primary_rg)</td></tr> <tr><td>47</td><td>parallelepiped</td><
solvent_sld, a_side, b_side, c_side)</td></tr> <tr><td>48</td><td>parallelepiped_xy</td>
solvent_sld, a_side, b_side, c_side, theta, phi, psi)</td></tr> <tr><td>49</td><td>pearl_nec
edge_separation, string_thickness, number_of_pearls, sld, string_sld, solvent_sld)</td></tr>
<tr><td>50</td><td>pearl_necklace_xy</td><td>(radius, edge_separation, string_thickness
number_of_pearls, sld, string_sld, solvent_sld)</td></tr> <tr><td>51</td><td>sphere</td>
solvent_sld, radius)</td></tr> <tr><td>52</td><td>sphere_xy</td><td>(sld,
solvent_sld, radius)</td></tr> <tr><td>53</td><td>star_polymer</td><td>(radius2,
arms)</td></tr> <tr><td>54</td><td>star_polymer_xy</td><td>(radius2, arms)</td></
<tr><td>55</td><td>stickyhardsphere</td><td>(effect_radius, volfraction, per-
turb, stickiness)</td></tr> <tr><td>56</td><td>stickyhardsphere_xy</td><td>(effect_radi
volfraction, perturb, stickiness)</td></tr> <tr><td>57</td><td>triaxial_ellipsoid</td><td>
solvent_sld, req_minor, req_major, rpolar)</td></tr> <tr><td>58</td><td>triaxial_ellipsoid
solvent_sld, req_minor, req_major, rpolar, theta, phi, psi)</td></tr> </table> <!DOC-
TYPE html> <html> <body> Available SasView sasmodels -
Extended parameters description <table border=1> <tr> <td>sasmodels
name</td> <td>Parameters </td> <td>Units</td> <td>Range</td> </tr> <tr>
<td>barbell</td>

```
<td> { 4, 1, 40, 20, 400 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td>barbell_xy</td>

```
<td> { 4, 1, 40, 20, 400, 60, 60 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td>
<td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td>
<td> </td> <td> </td> </tr> <tr> <td> </td> <td>Barbell
scattering length density</td> <td>4e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr>
<td> </td> <td>Solvent scattering length density</td> <td>1e-6/Å^2</td>
<td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Spherical bell radius</td>
<td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Cylindrical
bar radius</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td>
<td>Cylinder bar length</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td>
<td>In plane angle</td> <td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td>
<td>Out of plane angle</td> <td>degrees</td> <td>[-inf, inf]</td> </tr> <tr>

<td> </td> <td> </td> <td> </td> <td> </td> </tr>
<tr> <td>bcc</td>

```
<td> { 220, 0.06, 40, 4, 1 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td>bcc_xy</td>

```
<td> { 220, 0.06, 40, 4, 1, 60, 60, 60 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td>
<td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td>
<td> </td> <td> </td> </tr> <tr> <td> </td> <td>Nearest
neighbour distance</td> <td>Å</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td>
<td>Paracrystal distortion factor</td> <td> </td> <td>[-inf, inf]</td> </tr>
<tr> <td> </td> <td>Particle radius</td> <td>Å</td> <td>[0, inf]</td>
</tr> <tr> <td> </td> <td>Particle scattering length density</td> <td>1e-
6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Solvent scat-
tering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr>
<td> </td> <td>In plane angle</td> <td>degrees</td> <td>[-inf, inf]</td>
</tr> <tr> <td> </td> <td>Out of plane angle</td> <td>degrees</td>
<td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Out of plane angle</td>
<td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td> </td>
<td> </td> <td> </td> </tr> <tr> <td>capped_cylinder</td>

```
<td> { 4, 1, 20, 20, 400 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td>capped_cylinder_xy</td>

```
<td> { 4, 1, 20, 20, 400, 60, 60 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td>
<td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td>
<td> </td> <td> </td> </tr> <tr> <td> </td> <td>Cylinder
scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr>
<td> </td> <td>Solvent scattering length density</td> <td>1e-6/Å^2</td>
<td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Cylinder radius</td> <td>Å</td>
<td>[0, inf]</td> </tr> <tr> <td> </td> <td>Cap radius</td> <td>Å</td>
<td>[0, inf]</td> </tr> <tr> <td> </td> <td>Cylinder length</td> <td>Å</td>
<td>[0, inf]</td> </tr> <tr> <td> </td> <td>In plane angle</td> <td>degrees</td>
<td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Out of plane angle</td>
<td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td> </td>
<td> </td> <td> </td> </tr> <tr> <td>core_shell_cylinder</td>

```
<td> { 4, 4, 1, 20, 20, 400 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td>core_shell_cylinder_xy</td>

<td> { 4, 4, 1, 20, 20, 400, 60, 60 }</td>

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Cylinder core scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Cylinder shell scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Solvent scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Cylinder core radius</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Cylinder shell thickness</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Cylinder length</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>In plane angle</td> <td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Out of plane angle</td> <td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>cylinder</td>

<td> { 4, 1, 20, 400 }</td>

<td> </td> <td> </td> </tr> <tr> <td>cylinder_xy</td>

<td> { 4, 1, 20, 400, 60, 60 }</td>

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Cylinder scattering length density</td> <td>4e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Solvent scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Cylinder radius</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Cylinder length</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>In plane angle</td> <td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Out of plane angle</td> <td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>dab</td>

<td> { 50.0 }</td>

<td> </td> <td> </td> </tr> <tr> <td>dab_xy</td>

<td> { 50.0 }</td>

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>correlation length</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>ellipsoid</td>

<td> { 4, 1, 20, 400 }</td>

<td> </td> <td> </td> </tr> <tr> <td>ellipsoid_xy</td>

<td> { 4, 1, 20, 400, 60, 60 }</td>

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Ellipsoid scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Solvent scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Polar radius</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Equatorial radius</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>In plane angle</td> <td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Out of plane angle</td> <td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>fcc</td>

<td> { 220, 0.06, 40, 4, 1 }</td>

<td> </td> <td> </td> </tr> <tr> <td>fcc_xy</td>

<td> { 220, 0.06, 40, 4, 1, 60, 60, 60 }</td>

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Nearest neighbour distance</td> <td>Å</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Paracrystal distortion factor</td> <td> </td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Particle radius</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Particle scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Solvent scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>In plane angle</td> <td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Out of plane angle</td> <td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Out of plane angle</td> <td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>flexible_cylinder_ex</td>

<td> { 1000.0, 100.0, 20.0, 1.5, 1.0, 6.3 }</td>

<td> </td> <td> </td> </tr> <tr> <td>flexible_cylinder_ex_xy</td>

<td> { 1000.0, 100.0, 20.0, 1.5, 1.0, 6.3 }</td>

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Length of the flexible cylinder</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Kuhn length of the flexible cylinder</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Radius of the flexible cylinder</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Axis_ratio (major_radius/radius</td> <td> </td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Cylinder scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Solvent scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>gaussian_peak</td>

```
<td> { 0.05, 0.005 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td>gaussian_peak_xy</td>

```
<td> { 0.05, 0.005 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Peak position</td> <td>1/Å</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Peak width (standard deviation)</td> <td>1/Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>guinier</td>

```
<td> { 60.0 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td>guinier_xy</td>

```
<td> { 60.0 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Radius of Gyration</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>hardsphere</td>

```
<td> { 50.0, 0.2 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td>hardsphere_xy</td>

```
<td> { 50.0, 0.2 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>effective radius of hard sphere</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>volume fraction of hard spheres</td> <td> </td>

```
<td>[0, 0.74]</td>
```

</tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>HayterMSAsq</td>

```
<td> { 20.75, 19.0, 0.0192, 318.16, 0.0, 71.08 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td>HayterMSAsq_xy</td>

```
<td> { 20.75, 19.0, 0.0192, 318.16, 0.0, 71.08 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>effective radius of hard sphere</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>charge on sphere (in electrons)</td> <td>e</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>volume fraction of spheres</td> <td> </td>

```
<td>[0, 0.74]</td>
```

</tr> <tr> <td> </td> <td>temperature, in Kelvin, for Debye length calculation</td> <td>K</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>conc of salt, 1:1 electolyte, for Debye length</td> <td>M</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>dielectric constant of solvent (default water), for Debye length</td> <td> </td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>hollow_cylinder</td>

```
<td> { 30.0, 20.0, 400.0, 6.3, 1 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td>hollow_cylinder_xy</td>

```
<td> { 30.0, 20.0, 400.0, 6.3, 1, 90, 0 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Cylinder radius</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Hollow core radius</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Cylinder length</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Cylinder sld</td> <td>1/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Solvent sld</td> <td>1/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Theta angle</td> <td>degrees</td>

<td>[-360, 360]</td>

</tr> <tr> <td> </td> <td>Phi angle</td> <td>degrees</td>

<td>[-360, 360]</td>

</tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>lamellar</td>

<td> { 1, 6, 50 }</td>

<td> </td> <td> </td> </tr> <tr> <td>lamellar_xy</td>

<td> { 1, 6, 50 }</td>

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Layer scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Solvent scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Bilayer thickness</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>lamellarCaille</td>

<td> { 30.0, 20, 400.0, 0.1, 6.3, 1.0 }</td>

<td> </td> <td> </td> </tr> <tr> <td>lamellarCaille_xy</td>

<td> { 30.0, 20, 400.0, 0.1, 6.3, 1.0 }</td>

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>sheet thickness</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Number of layers</td> <td> </td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>d-spacing of Caille S(Q)</td> <td>Å</td>

<td>[0.0, inf]</td>

</tr> <tr> <td> </td> <td>Caille parameter</td> <td>1/Å^2</td>

<td>[0.0, 0.8]</td>

</tr> <tr> <td> </td> <td>layer scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Solvent scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>lamellarCailleHG</td>

<td> { 10, 2, 30, 40.0, 0.001, 0.4, 2.0, 6 }</td>

<td> </td> <td> </td> </tr> <tr> <td>lamellarCailleHG_xy</td>

<td> { 10, 2, 30, 40.0, 0.001, 0.4, 2.0, 6 }</td>

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Tail thickness</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>head thickness</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Number of layers</td> <td> </td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>d-spacing of Caille S(Q)</td> <td>Å</td>

<td>[0.0, inf]</td>

</tr> <tr> <td> </td> <td>Caille parameter</td> <td> </td>

<td>[0.0, 0.8]</td>

</tr> <tr> <td> </td> <td>Tail scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Head scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Solvent scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>lamellarFFHG</td>

<td> { 15, 10, 0.4, 3.0, 6 }</td>

<td> </td> <td> </td> </tr> <tr> <td>lamellarFFHG_xy</td>

<td> { 15, 10, 0.4, 3.0, 6 }</td>

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Tail thickness</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>head thickness</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Tail scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Head scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Solvent scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>lamellarPC</td>

<td> { 33.0, 20, 250.0, 0.0, 1.0, 6.34 }</td>

<td> </td> <td> </td> </tr> <tr> <td>lamellarPC_xy</td>

<td> { 33.0, 20, 250.0, 0.0, 1.0, 6.34 }</td>

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>sheet thickness</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Number of layers</td> <td> </td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>d-spacing of paracrystal stack</td> <td>Å</td>

<td>[0.0, inf]</td>

</tr> <tr> <td> </td> <td>d-spacing polydispersity</td> <td>Å</td>

<td>[0.0, inf]</td>

</tr> <tr> <td> </td> <td>layer scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Solvent scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>linear_pearls</td>

<td> { 80.0, 350.0, 3.0, 1.0, 6.3 }</td>

<td> </td> <td> </td> </tr> <tr> <td>linear_pearls_xy</td>

<td> { 80.0, 350.0, 3.0, 1.0, 6.3 }</td>

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Radius of the pearls</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Length of the string segment - surface to surface</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Number of the pearls</td> <td> </td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>SLD of the pearl spheres</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>SLD of the solvent</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>lorentz</td>

<td> { 50.0 }</td>

<td> </td> <td> </td> </tr> <tr> <td>lorentz_xy</td>

<td> { 50.0 }</td>

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Screening length</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>mass_fractal</td>

```
<td> { 10.0, 1.9, 100.0 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td>mass_fractal_xy</td>

```
<td> { 10.0, 1.9, 100.0 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Particle radius</td> <td>Å</td>

```
<td>[0.0, inf]</td>
```

</tr> <tr> <td> </td> <td>Mass fractal dimension</td> <td> </td>

```
<td>[1.0, 6.0]</td>
```

</tr> <tr> <td> </td> <td>Cut-off length</td> <td>Å</td>

```
<td>[0.0, inf]</td>
```

</tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>mass_surface_fractal</td>

```
<td> { 1.8, 2.3, 86.7, 4000.0 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td>mass_surface_fractal_xy</td>

```
<td> { 1.8, 2.3, 86.7, 4000.0 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Mass fractal dimension</td> <td> </td>

```
<td>[1e-16, 6.0]</td>
```

</tr> <tr> <td> </td> <td>Surface fractal dimension</td> <td> </td>

```
<td>[1e-16, 6.0]</td>
```

</tr> <tr> <td> </td> <td>Cluster radius of gyration</td> <td>Å</td>

```
<td>[0.0, inf]</td>
```

</tr> <tr> <td> </td> <td>Primary particle radius of gyration</td> <td>Å</td>

```
<td>[0.0, inf]</td>
```

</tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>parallelepiped</td>

```
<td> { 4, 1, 35, 75, 400 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td>parallelepiped_xy</td>

```
<td> { 4, 1, 35, 75, 400, 60, 60, 60 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parallelepiped scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Solvent scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Shorter side of the parallelepiped</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Second side of the parallelepiped</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Larger side of the parallelepiped</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>In plane angle</td> <td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Out of plane angle</td> <td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Rotation angle around its own c axis against q plane</td> <td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>pearl_necklace</td>

```
<td> { 80.0, 350.0, 2.5, 3, 1.0, 1.0, 6.3 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td>pearl_necklace_xy</td>

```
<td> { 80.0, 350.0, 2.5, 3, 1.0, 1.0, 6.3 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Mean radius of the chained spheres</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Mean separation of chained particles</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Thickness of the chain linkage</td>

<td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Mean number of pearls in each necklace</td> <td>none</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Scattering length density of the chained spheres</td> <td>Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Scattering length density of the chain linkage</td> <td>Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Scattering length density of the solvent</td> <td>Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>sphere</td>

<td> { 1, 6, 50 }</td>

<td> </td> <td> </td> </tr> <tr> <td>sphere_xy</td>

<td> { 1, 6, 50 }</td>

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Layer scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Solvent scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Sphere radius</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>star_polymer</td>

<td> { 100.0, 3 }</td>

<td> </td> <td> </td> </tr> <tr> <td>star_polymer_xy</td>

<td> { 100.0, 3 }</td>

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Ensemble radius of gyration squared of an arm</td> <td>Å</td>

<td>[0.0, inf]</td>

</tr> <tr> <td> </td> <td>Number of arms in the model</td> <td> </td>

<td>[1.0, 6.0]</td>

</tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>stickyhardsphere</td>

<td> { 50.0, 0.2, 0.05, 0.2 }</td>

<td> </td> <td> </td> </tr> <tr> <td>stickyhardsphere_xy</td>

```
<td> { 50.0, 0.2, 0.05, 0.2 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>effective radius of hard sphere</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>volume fraction of hard spheres</td> <td> </td>

```
<td>[0, 0.74]</td>
```

</tr> <tr> <td> </td> <td>perturbation parameter, epsilon</td> <td> </td>

```
<td>[0.01, 0.1]</td>
```

</tr> <tr> <td> </td> <td>stickiness, tau</td> <td> </td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td>triaxial_ellipsoid</td>

```
<td> { 4, 1, 20, 400, 10 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td>triaxial_ellipsoid_xy</td>

```
<td> { 4, 1, 20, 400, 10, 60, 60, 60 }</td>
```

<td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Parameters:</td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td>Ellipsoid scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Solvent scattering length density</td> <td>1e-6/Å^2</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Minor equitorial radius</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Major equatorial radius</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>Polar radius</td> <td>Å</td> <td>[0, inf]</td> </tr> <tr> <td> </td> <td>In plane angle</td> <td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Out of plane angle</td> <td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td>Out of plane angle</td> <td>degrees</td> <td>[-inf, inf]</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> </table>

```
<p> </p>
<p> </p>
```

Name convention: SasView - sasmodels <table border=1> <tr> <td>SasView name</td> <td>sasmodels name </td> <td>P(Q) </td> <td>S(Q)</td> <td>Multiply</td> <td>Multiplicity</td> </tr> <tr> <td>BarBell</td> <td>barbell </td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>BCCrystal</td> <td>bcc </td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>CappedCylinder</td> <td>capped_cylinder
</td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>CoreShellCylinder</td> <td>core_shell_cylinder
</td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>Cylinder</td> <td>cylinder </td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>DAB</td> <td>dab </td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>Ellipsoid</td> <td>ellipsoid </td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>FCCrystal</td> <td>fcc </td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>FlexCylEllipX</td> <td>flexible_cylinder_ex
</td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>PeakGauss</td> <td>gaussian_peak
</td>

```
<td>- </td>
<td> - </td>
```

<td>-</td> <td>-</td> </tr> <tr> <td>Guinier</td> <td>guinier </td>

```
<td>- </td>
<td> - </td>
```

<td>-</td> <td>-</td> </tr> <tr> <td>HardsphereStructure</td> <td>hardsphere
</td>

```
<td>- </td>
```

<td> Y </td> <td>-</td> <td>-</td> </tr> <tr> <td>HayterMSAStructure</td>
<td>HayterMSAsq </td>

```
<td>- </td>
```

<td> Y </td> <td>-</td> <td>-</td> </tr> <tr> <td>HollowCylinder</td>
<td>hollow_cylinder </td>

```
<td>- </td>
<td> - </td>
```

<td>-</td> <td>-</td> </tr> <tr> <td>Lamellar</td> <td>lamellar </td>

```
<td>- </td>
<td> - </td>
```

<td>-</td> <td>-</td> </tr> <tr> <td>LamellarPS</td> <td>lamellarCaille
</td>

```
<td>- </td>
<td> - </td>
```

<td>-</td> <td>-</td> </tr> <tr> <td>LamellarPSHG</td> <td>lamellarCailleHG
</td>

```
<td>- </td>
<td> - </td>
```

<td>-</td> <td>-</td> </tr> <tr> <td>LamellarFFHG</td> <td>lamellarFFHG
</td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>LamellarPCrystal</td> <td>lamellarPC
</td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>LinearPearls</td> <td>linear_pearls
</td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>Lorentz</td> <td>lorentz </td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>MassFractal</td> <td>mass_fractal
</td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>MassSurfaceFractal</td> <td>mass_surface_fractal
</td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>Parallelepiped</td> <td>parallelepiped
</td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>PearlNecklace</td> <td>pearl_necklace
</td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>Sphere</td> <td>sphere </td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>StarPolymer</td> <td>star_polymer
</td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> <tr> <td>StickyHSStructure</td> <td>stickyhardsphere
</td>

<td>- </td>

<td> Y </td> <td>-</td> <td>-</td> </tr> <tr> <td>TriaxialEllipsoid</td>
<td>triaxial_ellipsoid </td>

<td>- </td>
<td> - </td>

<td>-</td> <td>-</td> </tr> </table> </body> </html> MDOC_END

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
model_index		Index of the applied sasview model. Recompile instrument for changes to take effect.	21
model_scale		Global scale factor for scattering kernel. For systems without inter-particle interference, the form factors can be related to the scattering intensity by the particle volume fraction.	1.0
model_pars		Model parameters are given as a set of comma-separated values enclosed by {}, e.g. {60} for the model index 21 (the guinier model). Consult the sasview docs for further info.	{60,0,0,0,0,0,0,0,0,0,0
model_abs	1/m	Absorption cross section density at 2200 m/s	0.5
xwidth	m	horiz. dimension of sample, as a width	0

Name	Unit	Description	Default
yheight	m	vert . dimension of sample, as a height for cylinder/box	0
zdepth	m	depth of sample	0
radius	m	Outer radius of sample in (x,z) plane for cylinder/sphere	0
target_x	m	relative focus target position	0
target_y	m	relative focus target position	0
target_z	m	relative focus target position	6
target_index	1	Relative index of component to focus at, e.g. next is +1	0
focus_xw	m	horiz. dimension of a rectangular area	0
focus_yh	m	vert. dimension of a rectangular area	0
focus_aw	deg	horiz. angular dimension of a rectangular area	0
focus_ah	deg	vert. angular dimension of a rectangular area	0

Links

- Component source code found in file `SasView_model.comp`.
- http://www.sasview.org/sasview/user/models/model_functions.html

15.17. The `Guide_anyshape_r` McStas Component

OBSOLETE: use Guide_anyshape(file_coatings=1) instead. Reflecting surface (guide and mirror) with any shape, defined from an OFF file and a patch to allow m-, alpha- and W-values added per polygon face. Derived from Guide_anyshape / Guide_anyshape_r .

Identification

- **Author:** Emmanuel Farhi, adapted by Peter Link and Gaetano Mangiapia
- **Origin:** ILL/MLZ
- **Date:** August 4th 2010

Description

OBSOLETE: this component has been merged into Guide_anyshape. The per-face coating mode of Guide_anyshape_r(m=0, alpha=0, W=0, ...) is obtained with Guide_anyshape(geometry=..., file_coatings=1, R0=..., Qc=...) and the parameter override mode (any of m, alpha, W

non-zero) with Guide_anyshape(geometry=..., m=..., alpha=..., W=...) This version is kept only for backward compatibility of existing instruments.

This is a reflecting object component, derived from Guide_anyshape. Its shape, as well as m-, alpha- and W-values (IN THIS ORDER) for each face, are defined from an OFF file, given with its file name. The object size is set as given from the file, where dimensions should be in meters. The bounding box may be re-scaled by specifying xwidth,yheight,zdepth parameters. The object may optionally be centered when using 'center=1'.

If in input only the values of R0 and Qc are specified, leaving m, alpha and W all null, then Guide_anyshape_g will use the values of these last three parameters that are specified in the OFF file for each face: floating point based values may be provided for them. In case at least one among m, alpha and W is not null, then Guide_anyshape_g will use these values as input (common for all the faces), ignoring those specified in the OFF file

The complex OFF/PLY geometry option handles any closed non-convex polyhedra. It supports the OFF (incl. NOFF, COFF), PLY and STL file formats. Standard .OFF files may be generated from XYZ data using: qhull < coordinates.xyz Qx Qv Tv o > geomview.off or powercrust coordinates.xyz and viewed with geomview or java -jar jroff.jar (see below). The default size of the object depends of the OFF file data, but its bounding box may be resized using xwidth,yheight and zdepth. PLY geometry files are also supported.

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default
xwidth	m	Redimension the object bounding box on X axis is non-zero	0
yheight	m	Redimension the object bounding box on Y axis is non-zero	0
zdepth	m	Redimension the object bounding box on Z axis is non-zero	0
center	1	When set to 1, the object will be centered w.r.t. the local coordinate frame	0
transmit	1	When true, non reflected neutrons are transmitted through the surfaces, instead of being absorbed. No material absorption is taken into account though	0
R0	1	Low-angle reflectivity	0.99
Qc	$Å^{-1}$	Critical scattering vector	0.0219
alpha	Å	Slope of reflectivity	0

Name	Unit	Description	Default
m	1	m-value of material. Zero means completely absorbing.	0
W	$Å^{-1}$	Width of supermirror cut-off	0
geometry	str	Name of the OFF, PLY or STL file describing the guide geometry	0

Links

- Component source code found in file `Guide_anyshape_r.comp`.
- Geomview and Object File Format (OFF)
- Java version of Geomview (display only) jroff.jar
- qhull
- powercrust

15.18. The `MCPL_input_once` McStas Component

Source-like component that reads neutron state parameters from a MCPL-file one time.

Identification

- **Author:** Gregory S Tucker
- **Origin:** European Spallation Source ERIC
- **Date:** Sep 2024, reduced to an INHERIT of MCPL_input Oct 2026

Description

Source-like component that reads neutron state parameters from a MCPL-file one time.

Since Oct 2026 MCPL_input_once is MCPL_input with repeat_count=1 (and the default replicate_across_processes=0) forced: the file is shared between MPI processes and every event in the file is used exactly once. The parameters are inherited from MCPL_input; repeat_count and replicate_across_processes are ignored. See MCPL_input for the full documentation.

MCPL is short for Monte Carlo Particle List, and is a format for sharing events between e.g. MCNP(X), Geant4 and McStas.

When used with MPI, the file contents are shared between workers with each accessing approximately (#events in the file) / (#MPI nodes)

Example: MCPL_input_once(filename=voutput,v_smear=0.1,pos_smear=0.001,dir_smear=0.01)

Input parameters

Parameters in **boldface** are required; the others are optional.

Name	Unit	Description	Default

Links

- Component source code found in file `MCPL_input_once.comp`.

Input parameters

`MCPL_input_once` is defined as `DEFINE COMPONENT MCPL_input_once INHERIT MCPL_input` and takes the input parameters of `MCPL_input` (see the Misc chapter), but forces `repeat_count=1` and `replicate_across_processes=0`: each event in the MCPL file is used exactly once, also when running with MPI. Use `MCPL_input` in new instruments.

A. Polarisation in McStas

McStas provides a set of components for simulating neutron beam polarisation and the optical elements that manipulate it: polarising monochromators, mirrors and guides, magnetic-field regions (including spin flippers and spin-echo coils), and polarisation-sensitive monitors. This chapter originated as an appendix written by P. Christiansen documenting the first polarisation components (2006); it is now promoted to a full chapter and brought up to date with the current McStas 3.x component set and the `pol-lib` field/precession framework that underlies it.

We first describe the polarisation vector and how it is related to the neutron wavefunction (section A.1) and then the physics of the scattering processes relevant to polarisation (section A.2). Section A.3 describes in detail how McStas propagates the polarisation vector through magnetic fields – the numerical precession algorithm and the field functions available – since this underlies every magnetic-field component in the library. Section A.4 then documents the current polarisation components themselves, organised by role, and section A.5 lists example and test instruments, including a worked spin-echo example.

We rely heavily on the books [Lov84; Wil88] for the underlying physics, where the detailed calculations can be found.

The notation used here (and in [Wil88]) is P (scalar), $\mathbf{P}$ (vector), $\tilde{\mathbf{P}}$ (unit-vector), $\hat{\sigma}$ (operator), and $\hat{\boldsymbol{\sigma}}$ (vector of operators).

A.1. The Polarization Vector

The spin of the neutron is represented by an operator $\hat{\mathbf{s}}$ for which only a single component can be measured at one time. Each single measurement will give a value $\pm 1/2$, but if we could make a large number of measurements on the same neutron state, in each of the three axis directions, and then make the average we get $\langle \hat{\mathbf{s}} \rangle$. The polarization vector, $\mathbf{P}$, is then defined as:

$$\mathbf{P} = \frac{\langle \hat{\mathbf{s}} \rangle}{s}, \tag{A.1}$$

so that $-1 \leq |\mathbf{P}| \leq +1$

For a neutron beam which contains N neutrons, each with a polarization $\mathbf{P}_i$, the beam polarization is defined as:

$$\mathbf{P} = \frac{\sum_i \mathbf{P}_i}{N}. \tag{A.2}$$

If we have one common quantization direction (e.g. a magnetic field direction) each neutron will either be spin up, $\uparrow$, or spin down, $\downarrow$, and the polarization can be expressed as:

$$P = \frac{n_\uparrow - n_\downarrow}{n_\uparrow + n_\downarrow}, \tag{A.3}$$

where ν ($n_\downarrow$) is the number of neutrons with spin up (down).

For a given neutron the probability of the neutron being spin up, $P(\uparrow)$, is:

$$P(\uparrow) = \frac{n_\uparrow}{n_\uparrow + n_\downarrow} = \frac{n_\uparrow + (n_\downarrow - n_\downarrow)/2}{n_\uparrow + n_\downarrow} = \frac{1+P}{2}, \tag{A.4}$$

and $P(\downarrow) = 1 - P(\uparrow) = (1-P)/2$.

The expectation value of the 'spin' operator, $\hat{\boldsymbol{\sigma}}$, which can be expressed by the Pauli matrices, is the polarization vector $\mathbf{P}$, $\mathbf{P} = \langle\hat{\boldsymbol{\sigma}}\rangle \equiv \langle\chi|\hat{\boldsymbol{\sigma}}|\chi\rangle$. The most general form of the spin wave-function χ for a neutron (spin 1/2) is:

$$\chi = a\chi_\uparrow + b\chi_\downarrow, \tag{A.5}$$

where $\chi_\uparrow$ and $\chi_\downarrow$ are eigenfunction of $\hat{\sigma}^z$, and the complex coefficients a and b satisfy $|a|^2 + |b|^2 = 1$.

By calculation we find:

$$P_x = \langle\chi|\hat{\sigma}_x|\chi\rangle = 2\mathrm{Re}(a^*b) \tag{A.6}$$

$$P_y = \langle\chi|\hat{\sigma}_y|\chi\rangle = 2\mathrm{Im}(a^*b) \tag{A.7}$$

$$P_z = \langle\chi|\hat{\sigma}_z|\chi\rangle = |a|^2 - |b|^2 \tag{A.8}$$

This shows the relation of the polarization vector to the neutron wave function.

The neutron magnetic moment operator can be expressed in terms of $\hat{\boldsymbol{\sigma}}$, as:

$$\hat{\boldsymbol{\mu}}_\mathbf{n} = \mu_n\hat{\boldsymbol{\sigma}}, \tag{A.9}$$

which, as shown above, is related to the polarization vector.

In our simulation we represent the polarization by the vector $\mathbf{S} = (s_x, s_y, s_z)$ which is propagated through the different components so it has the correct relative orientation in each component. The probability for the spin to be parallel a given direction $\mathbf{n}$ is then:

$$P(\uparrow|\mathbf{n}) = \frac{1 + \mathbf{n}\cdot\mathbf{S}}{2}. \tag{A.10}$$

This equation (from [SD01]) is easy to understand. The average spin along $\mathbf{n}$ is $\mathbf{n}\cdot\mathbf{S}$ and the probability then follows from Eq. A.4.

For an unpolarized beam, $\mathbf{S} = \mathbf{0}$ and all directions are equally probable (50 %).

Note that in our approach we do not decide if the neutron is up or down after a given component, but instead keep track of as much information for as long as possible.

In the following we will use $\mathbf{P}$ to denote the polarization vector. The most important variables used are:

$\boldsymbol{\kappa}$	Scattering vector.
$\mathbf{P}$	Polarization before a component (ingoing).
$\mathbf{P}_\perp$	Polarization perpendicular to scattering vector, $\mathbf{P}_\perp = \tilde{\boldsymbol{\kappa}} \times (\mathbf{P} \times \tilde{\boldsymbol{\kappa}})$.
$\mathbf{P}'$	Polarization after a component (outgoing).
$\tilde{\boldsymbol{\eta}}$	Unit vector in direction of atomic spin ($\tilde{\boldsymbol{\eta}} \cdot \tilde{\mathbf{B}} = -1$ for a ferromagnet).
$F_N(\boldsymbol{\kappa})$	Unit cell nuclear structure factor.
$F_M(\boldsymbol{\kappa})$	Unit cell magnetic structure factor.

The unit cell nuclear structure factor is defined as:

$$F_N(\boldsymbol{\kappa}) = \sum_{\mathbf{d}} \exp(i\boldsymbol{\kappa} \cdot \mathbf{d}) \bar{b}_d, \tag{A.11}$$

where the $\mathbf{d}$ is the position of the d'th atom within the unit cell, and $\bar{b}_d$ is the average of b_d. In the simple case of a single atom Bravais crystal one finds $F_N(\boldsymbol{\kappa}) = \bar{b}$.

The unit cell magnetic structure factor is useful when the atoms in the crystal only have spin orbital angular momentum, and simple when the magnet is saturated (all spins are parallel or anti-parallel to *one* direction, $\sigma_d = \pm 1$). It is then given as:

$$F_M(\boldsymbol{\kappa}) = \gamma_n r_0 \sum_{\mathbf{d}} \exp(i\boldsymbol{\kappa} \cdot \mathbf{d}) \frac{1}{2} g_d F_d(\boldsymbol{\kappa}) \langle \hat{S}_d \rangle \sigma_d, \tag{A.12}$$

where $r_0 = \frac{\mu_0}{4\pi} \frac{e^2}{m_e} = 2.818 \times 10^{-15}$m, $g = 2$ is the Landé splitting factor, and $F_d(\boldsymbol{\kappa})$ is the magnetic form factor, which is the Fourier transform of the magnetization density (normalized so that $F_d(0) = 1$), and $\langle \hat{S}_d \rangle$ is the thermal average of the ordered atomic spin.

In the following the Debye-Weller factor ($\exp(-W_d)$) have been ignored in all cross sections.

A.1.1. Example: Magnetic fields

The magnetic moment operator of the neutron is $\hat{\boldsymbol{\mu}}_\mathbf{n} = \gamma_n \hat{\mathbf{s}}$, where $\gamma_n = 2\mu_n = -3.826$ is the gyromagnetic ratio (spin and magnetic moment is anti-parallel as for an electron) [1]

A magnetic field, $\mathbf{B}$, will exert a torque, $\boldsymbol{\sigma} = d\mathbf{s}/dt = (1/\gamma_n) d\boldsymbol{\mu}/dt$, on the neutron magnetic moment:

$$\frac{1}{\gamma_n} \frac{d\boldsymbol{\mu}}{dt} = \boldsymbol{\mu} \times \mathbf{B} \tag{A.13}$$

The magnetic moment μ can be related to the polarization as $\boldsymbol{\mu} = \gamma_n \mathbf{P}/2$, and inserting in Eq. A.13 we find:

$$\frac{d\mathbf{P}}{dt} = \gamma_n \mathbf{P} \times \mathbf{B} \tag{A.14}$$

In the simple case where $\mathbf{B} = (0, 0, B)$, we find the solution ([Wil88] p. 18) :

[1] Note that if we had used S (with values $S = \pm 1$) to define γ_n we would get $\gamma_n = -1.913$ which is also commonly used.

$$
\begin{aligned}
P_X(t) &= \cos(\omega_L t)P_X(0) - \sin(\omega_L t)P_Y(0) \\
P_Y(t) &= \sin(\omega_L t)P_X(0) + \cos(\omega_L t)P_Y(0) \\
P_Z(t) &= P_Z(0),
\end{aligned}
\tag{A.15}
$$

where $\omega_L = -\gamma_n B/\hbar$ is the Larmor frequency.

> The equations above were checked against the equations in the "polarimetrie neutronique" notes by Francis Tasset and found to be consistent. There can be sign differences between different publications depending on whether they use a right-handed (like e.g. McStas) or a left-handed (like e.g. NISP) coordinate system.

This closed-form solution is exactly what the **Pol_constBfield** and **Pol_FieldBox** components (section A.4) evaluate directly in one step for a spatially uniform field. For the general case of a spatially varying, rotating, or otherwise non-uniform field, McStas instead integrates the precession numerically – this general algorithm is described in section A.3, and is exactly the same physics applied piecewise over short enough intervals that the field can be treated as locally uniform.

A.2. Polarized Neutron Scattering

First we will give a short introduction to how calculations are done and then quote some results which are important for implementing the first McStas components.

All the potentials (nuclear, magnetic, and electric) we will be interested in can be written on the form:

$$
\hat{v} = \hat{\beta} + \hat{\boldsymbol{\alpha}} \cdot \hat{\boldsymbol{\sigma}} \tag{A.16}
$$

The first term does not affect the spin, while the second term can change the spin. Let us just remind here that:

$$
\begin{aligned}
\hat{\sigma}_x\chi_\uparrow = \chi_\downarrow, \quad \hat{\sigma}_y\chi_\uparrow = i\chi_\downarrow, \quad \hat{\sigma}_z\chi_\uparrow = \chi_\uparrow, \\
\hat{\sigma}_x\chi_\downarrow = \chi_\uparrow, \quad \hat{\sigma}_y\chi_\downarrow = -i\chi_\uparrow, \quad \hat{\sigma}_z\chi_\downarrow = -\chi_\downarrow.
\end{aligned}
\tag{A.17}
$$

So that the interaction proportional to $\hat{\sigma}_x$ and $\hat{\sigma}_y$ results in spin flips, while the interactions with $\hat{\sigma}_z$ conserves the spin.

It turns out to be smart to define a density matrix operator:

$$
\hat{\rho} = \chi\chi^\dagger = \begin{pmatrix} |a|^2 & ab^* \\ ba^* & |b|^2 \end{pmatrix} = \frac{1}{2}(\mathcal{I} + \mathbf{P} \cdot \hat{\boldsymbol{\sigma}}), \tag{A.18}
$$

where χ is the neutron wave function (Eq. A.5), and $\mathcal{I}$ is the unit matrix.

Using the density matrix the elastic cross section can be written as ([Lov84], Eq. 10.31):

$$
\frac{d\sigma}{d\Omega} = \mathrm{Tr}\hat{\rho}\hat{v}^\dagger\hat{v} = \sum_{\lambda,\lambda'} p_\lambda \mathrm{Tr}\hat{\rho}\langle\lambda|\hat{V}^\dagger(\boldsymbol{\kappa})|\lambda'\rangle\langle\lambda'|\hat{V}(\boldsymbol{\kappa})|\lambda\rangle\delta(E_\lambda - E_{\lambda'}), \tag{A.19}
$$

where $\hat{V}$ is the interaction potential and it is understood that the trace is to be taken with respect only to the neutron spin coordinates. The outgoing polarization is given as:

$$\mathbf{P}'\frac{d\sigma}{d\Omega} = \mathrm{Tr}\hat{\rho}\hat{v}^\dagger\hat{\boldsymbol{\sigma}}\hat{v} = \sum_{\lambda,\lambda'} p_\lambda \mathrm{Tr}\hat{\rho}\langle\lambda|\hat{V}^\dagger(\boldsymbol{\kappa})|\lambda'\rangle\hat{\boldsymbol{\sigma}}\langle\lambda'|\hat{V}(\boldsymbol{\kappa})|\lambda\rangle\delta(E_\lambda - E_{\lambda'}) \tag{A.20}$$

Inserting Eq. A.16 in Eq. A.19 and Eq. A.20 results in the two master equations for polarized neutron scattering:

$$\mathrm{Tr}\hat{\rho}\hat{v}^\dagger\hat{v} = \hat{\boldsymbol{\alpha}}^\dagger\cdot\hat{\boldsymbol{\alpha}} + \hat{\beta}^\dagger\hat{\beta} + \hat{\beta}^\dagger(\hat{\boldsymbol{\alpha}}\cdot\mathbf{P}) + (\hat{\boldsymbol{\alpha}}^\dagger\cdot\mathbf{P})\hat{\beta} + i\mathbf{P}\cdot(\hat{\boldsymbol{\alpha}}^\dagger\times\hat{\boldsymbol{\alpha}}), \tag{A.21}$$

and

$$\mathrm{Tr}\hat{\rho}\hat{v}^\dagger\hat{\boldsymbol{\sigma}}\hat{v} = \hat{\beta}^\dagger\hat{\boldsymbol{\alpha}}+\hat{\boldsymbol{\alpha}}^\dagger\hat{\beta}+\hat{\beta}^\dagger\hat{\beta}\mathbf{P}+\hat{\boldsymbol{\alpha}}^\dagger(\hat{\boldsymbol{\alpha}}\cdot\mathbf{P})+(\hat{\boldsymbol{\alpha}}^\dagger\cdot\mathbf{P})\hat{\boldsymbol{\alpha}}-\mathbf{P}(\hat{\boldsymbol{\alpha}}^\dagger\cdot\hat{\boldsymbol{\alpha}})-i\hat{\boldsymbol{\alpha}}^\dagger\times\hat{\boldsymbol{\alpha}}+i\hat{\beta}^\dagger(\hat{\boldsymbol{\alpha}}\times\mathbf{P})+i(\mathbf{P}\times\hat{\boldsymbol{\alpha}}^\dagger)\hat{\beta}. \tag{A.22}$$

Based on these two equations and the interaction potentials all the results presented in the following are derived in [Lov84].

A.2.1. Example: Nuclear scattering

The nuclear scattering potential for a crystal is:

$$\hat{V}_N(\boldsymbol{\kappa}) = \sum_{\mathbf{l},\mathbf{d}} \exp(i\boldsymbol{\kappa}\cdot\mathbf{R}_{ld})(A_{ld} + \frac{1}{2}B_{ld}\hat{\boldsymbol{\sigma}}\cdot\hat{\mathbf{I}}_{ld}), \tag{A.23}$$

so that

$$\begin{aligned}\hat{\boldsymbol{\alpha}} &= \sum_{\mathbf{l},\mathbf{d}} \exp(i\boldsymbol{\kappa}\cdot\mathbf{R}_{ld})\frac{1}{2}B_{ld}\hat{\mathbf{I}}_{ld}\\ \hat{\beta} &= \sum_{\mathbf{l},\mathbf{d}} \exp(i\boldsymbol{\kappa}\cdot\mathbf{R}_{ld})A_{ld},\end{aligned} \tag{A.24}$$

where $\hat{\mathbf{I}}$ is the nuclear spin operator and the constants A and B are related to the nuclear scattering lengths b^+ and b^- as $A = ((I+1)b^+ + Ib^-)/(2I+1)$ and $B = (b^+ + b^-)/(2I+1)$.

To calculate the polarization cross section and outgoing polarization we have to average over the nuclear spin (which we assume is random oriented), so that terms linear in $\hat{\boldsymbol{\alpha}}$ (three last terms in Eq. A.21) disappears. The scattering cross section ends up being (see [Lov84] p. 159):

$$\frac{d\sigma}{d\Omega} = \sum_{\mathbf{l},\mathbf{d},\mathbf{l}',\mathbf{d}'} \exp(i\boldsymbol{\kappa}\cdot(\mathbf{R}_{ld} - \mathbf{R}_{l'd'}))(|\overline{A}_d|^2 + \delta_{\mathbf{l},\mathbf{l}'}\delta_{\mathbf{d},\mathbf{d}'}[\overline{|A_d|^2} - |\overline{A}_d|^2 + \frac{1}{4}\overline{|B_d|^2}I_d(I_d+1)]) \tag{A.25}$$

where the first term is the coherent cross-section and the second term is the site-incoherent cross-section. Both terms are independent of $\mathbf{P}$ as expected for a system without a preferred internal direction.

The polarization in the final state is:

$$\mathbf{P}'\frac{d\sigma}{d\Omega} = \sum_{\mathbf{l},\mathbf{d},\mathbf{l}',\mathbf{d}'} \exp(i\boldsymbol{\kappa}\cdot(\mathbf{R}_{ld}-\mathbf{R}_{l'd'}))\mathbf{P}'(|\overline{A}_d|^2+\delta_{\mathbf{l},\mathbf{l}'}\delta_{\mathbf{d},\mathbf{d}'}[\overline{|A_d|^2}-|\overline{A}_d|^2-\frac{1}{12}\overline{|B_d|^2}I_d(I_d+1)]) \tag{A.26}$$

Comparing Eq. A.25 and Eq. A.26 we find that: 1) The nuclear coherent polarization is the same as the initial polarization. 2) The same is true for the incoherent scattering due to the random isotope distribution. 3) The nuclear incoherent scattering due to the random nuclear spin orientations has polarization $\mathbf{P}' = -1/3\mathbf{P}$ (for a random nuclear spin the associated Pauli matrix will 2/3 of the time point in the direction of $\hat{\sigma}_x$ and $\hat{\sigma}_y$ which according to Eq. A.17 flips the spin).

For Vanadium, where there is only one isotope and coherent scattering is negligible, we find $\mathbf{P}' = -1/3\mathbf{P}$. There is however one catch. If the probability for multiple scattering is large one has to take into account that after two scattering one has: $\mathbf{P}'(2) = 1/9\mathbf{P}$, and so forth. The average polarization after a thick vanadium target is therefore a sum of different contributions.

A.2.2. Example: Polarizing Monochromator and Guides

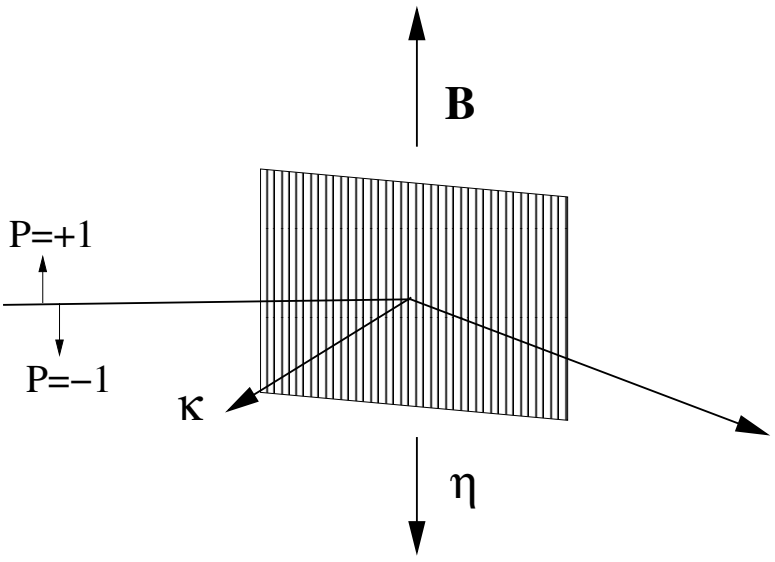


Figure A.1.: Principle and geometry of a polarizing monochromator.

In a polarized monochromator and polarizing guides we have a ferromagnetic crystal in an external magnetic field. The scattering potential is now both nuclear (no internal direction) and magnetic (internal direction), so in general the outgoing polarization can be quite complex. However, as illustrated in Figure A.1, the typical setup has many geometrical constraints: $\tilde{\boldsymbol{\eta}} \cdot \tilde{\boldsymbol{\kappa}} = 0$, $\tilde{\boldsymbol{\eta}} \cdot \mathbf{P}_\perp = \tilde{\boldsymbol{\eta}} \cdot \mathbf{P}$, and $\boldsymbol{\kappa} \times (\tilde{\boldsymbol{\eta}} \times \boldsymbol{\kappa}) = \tilde{\boldsymbol{\eta}}$, which simplifies the problem.

In [Lov84] the calculation for a centrosymmetric ferromagnetic crystal is done, and inserting the constraints above one finds ([Lov84], Eq. 10.96 and Eq. 10.110):

$$d\sigma/d\Omega = F_N(\boldsymbol{\kappa})^2 + 2F_N(\boldsymbol{\kappa})F_M(\boldsymbol{\kappa})(\mathbf{P} \cdot \tilde{\boldsymbol{\eta}}) + F_M(\boldsymbol{\kappa})^2 \tag{A.27}$$

$$\mathbf{P}' d\sigma/d\Omega = \mathbf{P}[F_N(\boldsymbol{\kappa})^2 - F_M(\boldsymbol{\kappa})^2] + \tilde{\boldsymbol{\eta}}[2F_N(\boldsymbol{\kappa})F_M(\boldsymbol{\kappa}) + 2(\tilde{\boldsymbol{\eta}} \cdot \mathbf{P})F_M(\boldsymbol{\kappa})^2] \tag{A.28}$$

> NB! Note that in [Wil88] Eq. 2.2.25 there is a minus in front of the second term in Eq. A.27. We have not been able to understand this discrepancy, which is probably due to notation. Most other authors agree with the minus in front of the second term (e.g. Squires and Francis Tasset).

For a beam which is initially unpolarized we find the outgoing polarization to be:

$$\mathbf{P}' = \frac{\tilde{\boldsymbol{\eta}} 2F_N(\boldsymbol{\kappa})F_M(\boldsymbol{\kappa})}{d\sigma/d\Omega} = \frac{2F_N(\boldsymbol{\kappa})F_M(\boldsymbol{\kappa})}{F_N(\boldsymbol{\kappa})^2 + F_M(\boldsymbol{\kappa})^2}\tilde{\boldsymbol{\eta}}, \tag{A.29}$$

so that the beam is fully polarized along $\tilde{\boldsymbol{\eta}}$ if $F_N(\boldsymbol{\kappa}) = \pm F_M(\boldsymbol{\kappa})$.

What we use to characterize the polarizing monochromator in practice is not $F_N(\boldsymbol{\kappa})$ and $F_M(\boldsymbol{\kappa})$, but instead the reflection probabilities $R_\uparrow$ and $R_\downarrow$ (for the reflection of interest).

If we assume that the reflection probabilities are directly proportional to the cross sections (with proportionality constant k), i.e., $R_\uparrow = kd\sigma/d\Omega(\mathbf{P} = +\tilde{\boldsymbol{\eta}})$ and $R_\downarrow = kd\sigma/d\Omega(\mathbf{P} = -\tilde{\boldsymbol{\eta}})$ then we can use Eq. A.27 to determine $F_N(\boldsymbol{\kappa})$ and $F_M(\boldsymbol{\kappa})$:

$$R_\uparrow = k(F_N(\boldsymbol{\kappa}) + F_M(\boldsymbol{\kappa}))^2, \tag{A.30}$$

$$R_\downarrow = k(F_N(\boldsymbol{\kappa}) - F_M(\boldsymbol{\kappa}))^2. \tag{A.31}$$

The values of $\sqrt{k}F_N(\boldsymbol{\kappa})$ and $\sqrt{k}F_M(\boldsymbol{\kappa})$ are then between -1 and +1 and unit less like the reflection probabilities. In the following we ignore k and just talk about $F_N(\boldsymbol{\kappa})$ and $F_M(\boldsymbol{\kappa})$.

In principle there are four solutions for $F_N(\boldsymbol{\kappa})$ and $F_M(\boldsymbol{\kappa})$, so in the code we currently choose the values where $F_N(\boldsymbol{\kappa}) + F_M(\boldsymbol{\kappa}) = +\sqrt{R_\uparrow}$ and $F_N(\boldsymbol{\kappa}) - F_M(\boldsymbol{\kappa}) = +\sqrt{R_\downarrow}$ (so that $F_N(\boldsymbol{\kappa}) > 0$ and $F_N(\boldsymbol{\kappa}) > F_M(\boldsymbol{\kappa})$). We then find:

$$F_N(\boldsymbol{\kappa}) = \frac{\sqrt{R_\uparrow} + \sqrt{R_\downarrow}}{2}, \tag{A.32}$$

$$F_M(\boldsymbol{\kappa}) = \frac{\sqrt{R_\uparrow} - \sqrt{R_\downarrow}}{2}. \tag{A.33}$$

When $F_N(\boldsymbol{\kappa})$ and $F_M(\boldsymbol{\kappa})$ are determined from these equations, Eq. A.27 and Eq. A.28 can easily be used to handle any situation.

This solution is both used for monochromators and guides.

It is not clear that this solution is correct. If we make a simple example with $R_\uparrow = 1$ and $R_\downarrow = 0.25$ then we could in principle have four solutions, but let us just quote the two where $F_N(\boldsymbol{\kappa})$ is positive since the last two are found by inserting a minus before all the solutions and this does not change the physics. The two solutions are $F_N(\boldsymbol{\kappa}) = 0.75, F_M(\boldsymbol{\kappa}) = 0.25$ and $F_M(\boldsymbol{\kappa}) = 0.75, F_N(\boldsymbol{\kappa}) = 0.25$. All solutions gives the same cross section, but if the incoming beam is polarized (and only then) the outgoing beam will have two different polarization values, since $\mathbf{P}[F_N(\boldsymbol{\kappa})^2 - F_M(\boldsymbol{\kappa})^2]$ and $\tilde{\boldsymbol{\eta}} 2(\tilde{\boldsymbol{\eta}} \cdot \mathbf{P}) F_M(\boldsymbol{\kappa})^2$ are different for the two solutions. It seems that one needs some additional information to choose between the two solutions.

> NB! The simplifying geometry shown in Figure A.1 only applies for the sides of the guide wall and not the top and bottom (assuming that the magnetizing field is pointing up or down), so there another set of equations should really be used.

The same physics could also be used for a polarizing powder or single crystal sample if $F_N(\boldsymbol{\kappa})$ and $F_M(\boldsymbol{\kappa})$ can be calculated with some other program, but one would have to use the general form of Eq. A.27 and Eq. A.28 without the simplifying geometrical constraints for monochromators and guides.

A.3. Magnetic fields: the precession algorithm and field framework

All magnetic-field-related polarisation components in McStas share a common run-time library, `pol-lib` (`mccode/nlib/share/pol-lib.c/.h`), which provides the spin-precession integrator and a small set of built-in field functions. A separate library, `ref-lib`, provides the standard supermirror reflectivity function (`StdReflecFunc`, section A.2.2) used by the polarising mirrors and guides.

A.3.1. The magnetic field stack

Rather than each field-producing component computing precession locally and independently, `pol-lib` maintains a small per-particle *stack* of active magnetic fields, `_particle->mcMagnet`. A component that defines a field region (e.g. **Pol_Bfield**) *pushes* a description of its field onto this stack when a particle enters the region, using

```
mcmagnet_push(_particle, field_type, &rotation, &position, stopbit,
params[8])
```

where `field_type` selects one of the built-in field functions (section A.3.3), `rotation`/`position` record the field region's placement in the lab frame (so the field can later be evaluated in the region's own local coordinates regardless of where in the instrument it sits), and

`params` holds up to 8 doubles of field-specific parameters (amplitude, extent, ...). A companion **Pol_Bfield_stop** component calls `mcmagnet_pop()` to remove the field from the stack once the particle leaves the region.

Because this is a stack rather than a single active field, **field regions may be nested or overlap**: any component placed between a `Pol_Bfield`/`Pol_Bfield_stop` pair – including another, independent field region – experiences the vector sum of every field currently on the stack (evaluated by `mcmagnet_get_field()`, which walks the whole stack, transforms each field into the lab frame, and adds the contributions). A field region may also be declared *closed* (`concentric=0`), in which case it pushes and pops its own field internally and no other components may be placed inside it – this is the simpler, non-nestable mode used when a single self-contained field region is all that is needed.

Whenever a particle is propagated through a region with an active field (technically: whenever `PROP_DT` is called while `_particle->mcMagnet` is non-`NULL`), the propagation macro transparently invokes the precession integrator described next, so that ordinary component code (slits, monitors, samples placed inside a field region) does not need to know anything about polarisation – the spin precesses correctly regardless of what else is happening to the particle during that propagation step.

A.3.2. Numerical spin precession: `SimpleNumMagnetPrecession`

The general-purpose precession routine, `SimpleNumMagnetPrecession()`, integrates Eq. A.14 numerically along the particle's straight-line trajectory (gravity is *not* accounted for inside the field itself) for a total propagation time dt, using an *adaptive-step* scheme:

1. The field $\mathbf{B}_{\text{start}}$ is evaluated at the particle's current position (by querying the whole field stack, or – for the self-contained field components – a single field function).

2. A trial sub-step of length `mc_pol_initial_timestep` (default 10^{-5} s, capped at the remaining dt) is proposed, and the field $\mathbf{B}_{\text{end}}$ is evaluated at the particle's position at the end of that trial step.

3. If the angle between $\mathbf{B}_{\text{start}}$ and $\mathbf{B}_{\text{end}}$ exceeds a threshold, `mc_pol_angular_accuracy` (default 1°), the trial step is halved and re-evaluated; this repeats until the field direction changes by less than the threshold over the sub-step (or the step size underflows to `FLT_EPSILON`). This is what makes the scheme *adaptive*: in a slowly-varying or uniform field, the integrator quickly settles on large sub-steps (up to the 10^{-5} s cap), while in a rapidly rotating field (e.g. the `rotating` or `majorana` functions of section A.3.3) it automatically refines the step until the field is well approximated as locally uniform.

4. Once an acceptable sub-step is found, the particle position and time are advanced by that sub-step, and the spin is precessed *about the mean field over the sub-step*, $\bar{\mathbf{B}} = \frac{1}{2}(\mathbf{B}_{\text{start}} + \mathbf{B}_{\text{end}})$, by the angle

$$\phi = \left(|\bar{\mathbf{B}}|\,\delta t\,\omega_L\right) \bmod 2\pi, \qquad \omega_L = -2\pi \times 29.16\ \text{MHz/T},$$

matching Eq. A.15 applied over the sub-step, using a generic axis-angle rotation of (s_x, s_y, s_z) about $\bar{\mathbf{B}}$ (rather than the special-case z-axis form of Eq. A.15, since $\bar{\mathbf{B}}$ may point in any direction).

5. This is repeated, consuming the remaining dt sub-step by sub-step, until the whole requested propagation time has been precessed over.

Positions/velocities/spin are tracked internally in the lab frame (fields are pushed to the stack together with the rotation/position needed to transform into and out of each field region's local frame) and the final spin is transformed back into the calling component's local frame before being returned. **Pol_tabled_field** (section A.4) re-implements this identical algorithm internally (rather than going through the shared field stack), so that it can query its own tabulated/ interpolated field directly.

Two component-specific accuracy parameters, exposed via `mc_pol_set_timestep()` and `mc_pol_set_angular_accuracy()`, allow the initial sub-step size and the angular-accuracy threshold to be tuned instrument-wide if the defaults are not adequate for a particular field geometry.

A.3.3. Available field functions

The field pushed onto the stack by **Pol_Bfield** is one of the following built-in functions, selected by the integer `field_type` parameter (`Pol_FieldBox` and `Pol_constBfield` always use the constant-field case, evaluated in closed form rather than via the stack):

1 – constant $\mathbf{B} = (B_x, B_y, B_z)$, uniform throughout the region.

2 – rotating $\mathbf{B}$ starts as $(0, B_y, 0)$ at the entrance and rotates smoothly to $(B_y, 0, 0)$ at the far end (over the region's `zdepth`), following $B_x = B\sin(\frac{\pi}{2}z/L)$, $B_y = B\cos(\frac{\pi}{2}z/L)$.

3 – majorana a large component along x decreases linearly from $+B_x$ to $-B_x$ across the region (passing through zero at the centre – the classic "Majorana flip" geometry for studying spin-flip loss at a field zero-crossing), plus a small, constant transverse component along y.

6 – gradient linear interpolation of the full field vector between two given endpoint vectors $\mathbf{B}_1$ (at $z = -L/2$) and $\mathbf{B}_2$ (at $z = +L/2$) – a generalisation of the majorana case to an arbitrary field-vector gradient.

-1 – tabled used internally by **Pol_tabled_field**: the field is not one of the analytic functions above but is instead interpolated (via `interpolation-lib`, choosing between a `kdtree` or `regular`-grid interpolator) from an externally supplied point cloud file of (x, y, z, B_x, B_y, B_z) rows – this is the route to use for a realistic, measured or finite-element-computed field map.

4 (MSF) and 5 (RF) reserved identifiers for a modulated/RF spin flipper field; not implemented in the current release (a radio-frequency field is more commonly modelled explicitly via a rotating-frame field function or, for an idealised flipper, by simply mirroring the spin vector, see **Pol_SF_ideal** below).

New field functions can be added by writing a function with the signature of e.g. `const_magnetic_field()` in `pol-lib.c` and adding a case to `magnetic_field_dispatcher()`; per the file header, this requires modifying the shared library rather than being an end-user-facing extension point.

A.4. Polarisation components in McStas

The components below can be grouped by role:

- **Setting and analysing polarisation** (unphysical, for testing/idealised use).
- **Polarising monochromators, mirrors and guides** (physical, reflectivity-based).
- **Magnetic field regions** (precession, using the framework of section A.3).
- **Polarisation-sensitive monitors**.
- **Samples** affecting polarisation.

As of McStas 3.x these are found in the `optics` and `monitors` component categories (not a dedicated "polarisation" category); a component is polarisation-aware precisely when its `TRACE` section reads and/or writes the particle's spin state (s_x, s_y, s_z).

A.4.1. Setting and analysing polarisation

- **Set_pol:** An unphysical, zero-size component (drawn like an `Arm`) used to impose a polarisation state, in one of four modes selected by `randomOn` and `normalize`: (i) hard-code the polarisation to (p_x, p_y, p_z); (ii) set it to a random vector *on* the unit sphere (fully polarised, random direction); (iii) set it to a random vector *within* the unit sphere (random direction and degree of polarisation); (iv) hard-code the *direction* of (p_x, p_y, p_z) but normalise it to unit polarisation.
- **PolAnalyser_ideal:** An unphysical, ideal polarisation analyser. Given an analysis direction $\mathbf{m} = (m_x, m_y, m_z)$ ($|\mathbf{m}| \leq 1$ for an imperfect analyser), it reduces the neutron weight by the projection probability $\frac{1}{2}(1 + \mathbf{S} \cdot \mathbf{m})$ and collapses the outgoing spin to $\mathbf{m}$, or absorbs the ray if that probability is zero or negative.
- **Pol_SF_ideal:** An idealised spin flipper: within an absorbing box, the polarisation vector is mirrored through the plane through the origin with normal (n_x, n_y, n_z) ($\mathbf{S}' = \mathbf{S} - 2(\mathbf{S} \cdot \hat{\mathbf{n}})\hat{\mathbf{n}}$); rays missing the box pass through untouched. This is the natural building block for a spin-echo instrument's π-flip coils (section A.5.4) when an idealised, 100%-efficient flipper is sufficient.

A.4.2. Polarising monochromators, mirrors and guides

These all implement the physics of section A.2.2 (Eqs. A.27–A.33): given reflectivities for spin up and down ($R_\uparrow$, $R_\downarrow$, either as `{R0,Qc,alpha,m,W}` parametrised curves via `ref-lib`'s `StdReflecFunc`, or as tabulated reflectivity files), the shared helper functions `GetMonoPolFNFM`, `GetMonoPolRefProb`, `SetMonoPolRefOut` and `SetMonoPolTransOut` (all in `pol-lib`) compute F_N, F_M, the reflection probability, and the outgoing polarisation for both the reflected and transmitted branches. Since a 2024 revision (E.B. Knudsen, P. Willendrup, H. Lee), all of these components treat a polarising reflection/transmission event as a projective quantum measurement along the mirror's quantisation axis: the in-plane spin components are explicitly zeroed ($s_x = s_z = 0$) at the point of interaction, before the new s_y is set by `SetMonoPolRefOut`/ `SetMonoPolTransOut` – reflecting that only the component of polarisation along the mirror's up direction survives a measurement-like interaction of this kind.

- **Monochromator_pol:** A flat, infinitely thin mosaic monochromator/analyser crystal (Gaussian mosaic and d-spread, billiard-ball reflection), the direct polarising analogue of **Monochromator_flat**, parametrised by `Rup`/`Rdown` (or `Q`/`DM` for the lattice spacing).

- **Pol_mirror:** A flat, infinitely thin, non-absorbing polarising mirror in the y-z plane, reflecting and/or transmitting according to `p_reflect` (use -1 to sample physically from the mirror's own reflectivity, or a fixed value to force a chosen reflect/transmit statistics split while re-weighting correctly).

- **Pol_bender:** A curved, multi-slit polarising bender (based on **Guide_curved**), with independently specifiable reflectivities for each of the four walls (top/bottom/left/right) and gravity support.

- **Pol_guide_mirror** / **Pol_guide_vmirror:** Straight rectangular guides with one (`_mirror`) or two, V-shaped (`_vmirror`) polarising supermirrors sitting on the diagonal(s) inside an otherwise ordinary (optionally non-polarising) guide. Setting the up and down reflectivities of the diagonal mirror equal turns either component into an unpolarised *frame-overlap mirror* instead – a deliberate dual use of the same geometry.

A.4.3. Magnetic field components

See section A.3 for the shared precession algorithm and field functions used by these.

- **Pol_Bfield** / **Pol_Bfield_stop:** The general-purpose field region, supporting all of the field functions of section A.3.3 and a box, cylindrical, spherical, or window-shaped extent. Used together as a `concentric` pair (`Pol_Bfield` pushes the field, other components may be placed freely inside, `Pol_Bfield_stop` pops it), or standalone as a closed region (`concentric=0`) that nothing else may be placed inside.

- **Pol_FieldBox:** A simple, self-contained, non-nestable box with a single uniform field (B_x, B_y, B_z), evaluated with a direct, single-step closed-form precession (Eq. A.15) rather than the general stack/integrator – the lightest-weight choice for a plain guide field or spin-flip coil.

- **Pol_tabled_field:** A field region (box/cylinder/sphere, or an arbitrary OFF/PLY-geometry volume) whose field is interpolated from an externally supplied point-cloud file, using the same adaptive-step precession algorithm re-implemented internally (section A.3.2). The interpolation backend (`kdtree` or `regular`-grid) can be chosen explicitly or left to an automatic CPU/GPU-appropriate default.

- **Pol_constBfield:** A rectangular, closed (non-nestable) box with a constant field along its local y-axis, evaluated with the closed-form solution of Eq. A.15 directly (no field stack). Convenient for guide fields and simple spin flippers: rather than specifying B directly, `fliplambda`/`flipangle` let the field be specified as "the field that flips a neutron of this wavelength by this angle over this component's length" – exactly the `GetConstantField()` helper of `pol-lib`.

A.4.4. Polarisation-sensitive monitors

- **Pol_monitor:** Measures the projection of the polarisation onto a user-defined direction $\mathbf{m} = (m_x, m_y, m_z)$, i.e. $\mathbf{m} \cdot \mathbf{S}$, integrated over all detected rays.

- **PolLambda_monitor:** As **Pol_monitor**, resolved as a function of wavelength λ.

- **MeanPolLambda_monitor:** As **PolLambda_monitor**, but reporting the *mean* polarisation projection per wavelength bin (rather than the summed/integrated signal) – the natural monitor for a spin-echo polarisation-vs.-parameter scan (section A.5.4).

- **PSD_monitor_4PI_spin:** A spherical, 4π position-sensitive monitor that additionally records the mean polarisation projection per pixel, for mapping how polarisation varies with scattering angle.

A.4.5. Samples and polarisation

Few McStas sample components currently propagate polarisation explicitly. The generic **Incoherent** component (the modern, general-purpose replacement for the historical, now-obsolete **V_sample**) can be parametrised to reproduce the classic depolarising-incoherent-scatterer result of section A.2's nuclear-scattering example ($\mathbf{P}' = -\frac{1}{3}\mathbf{P}$ per single incoherent scattering event, from the random projection of nuclear spin onto $\hat{\sigma}_x$ and $\hat{\sigma}_y$); as noted there, this simple factor is only exact for a single scattering event, and multiple scattering must be handled with some care (consecutive depolarisation by $(-\frac{1}{3})^n$ is only valid if each scattering order can be isolated). Magnetic (Bragg) scattering from an ordered magnetic structure, using the magnetic structure factor F_M of section A.1, is

not yet implemented as a general sample component; **Single_crystal** and **PowderN** presently model only nuclear scattering and do not read or write the spin state.

A.5. Test and example instruments

Every instrument below is found in the `Tests_polarization` category of the example library (browsable via `mcgui`'s "New from Template" menu, or `mcdoc`).

A.5.1. Component-level regression tests

These verify a polarising component reduces correctly to its non-polarising counterpart, or exercise one component's specific physics in isolation: *Test_Pol_Bender*, *Test_Pol_FieldBox*, *Test_Pol_Guide_mirror*, *Test_Pol_Guide_Vmirror*, *Test_Pol_Mirror*, *Test_Pol_MSF*, *Test_Pol_SF_ideal*, *Test_Pol_Set*, *Test_Pol_Tabled*, *Test_pol_ideal*, *Test_Magnetic_Constant*, *Test_Magnetic_Majorana*, and *Test_Magnetic_Rotation* (illustrating, respectively, the constant, majorana, and rotating field functions of section A.3.3).

A.5.2. Test_Pol_TripleAxis

A complete example of a polarised triple-axis spectrometer: polarising monochromator and analyser, a vanadium (depolarising incoherent) sample, and a spin flipper – illustrating how the individual components of section A.4 combine into a real instrument.

A.5.3. He3_spin_filter

Illustrates the use of a ^{3}He spin filter cell for beam or analysis-side polarisation.

A.5.4. A spin-echo example: SE_example

SE_example (and its variant *SE_example2*, both written for the 2010 PNCMI school in Delft) is a mock-up transmission spin-echo instrument, and a compact illustration of most of the machinery described in this chapter working together:

- **Pol_mirror** polarises the incoming beam and, symmetrically, analyses the scattered beam.
- **Pol_Bfield** regions define the two (oppositely-signed) precession guide fields on either side of the sample position, exactly the nested/concentric field-stack usage of section A.3.1 (other components – the sample, monitors – sit *inside* each field region, between its `Pol_Bfield`/`Pol_Bfield_stop` pair).
- **MeanPolLambda_monitor** records the mean polarisation projection as a function of wavelength – the classic spin-echo readout, whose oscillation as a small guide-field perturbation `dBz` is scanned traces out the echo signal.

A typical scan takes the form

```
mcrun SE_example.instr dBz=-0.0001,0.0001 -N41 -n1e5
```

sweeping a small field imbalance `dBz` between the two precession regions and recording the resulting polarisation echo via **MeanPolLambda_monitor**. The instrument's own parameters (`POL_ANGLE`, `Bguide`, `Bflip`, `dBz`, `Lam`, `dLam`) let the polariser/analyser angle, the guide- and flipper-field strengths, and the source wavelength band be adjusted independently, and an optional incoherent scatterer (`SAMPLE=1`) can be inserted to study the effect of depolarising sample scattering on the echo.

B. The Union components

The **Union components** are a set of McStas components, originally developed by Mads Bertelsen [Ber17; BJL17], that take a fundamentally different approach to sample and sample-environment simulation than ordinary McStas components. This chapter explains the concepts behind the Union components and how to use them from an instrument file; the full parameter reference for every individual Union component is auto-generated from the component headers and found in the Component Manual's Union chapter.

B.1. Motivation

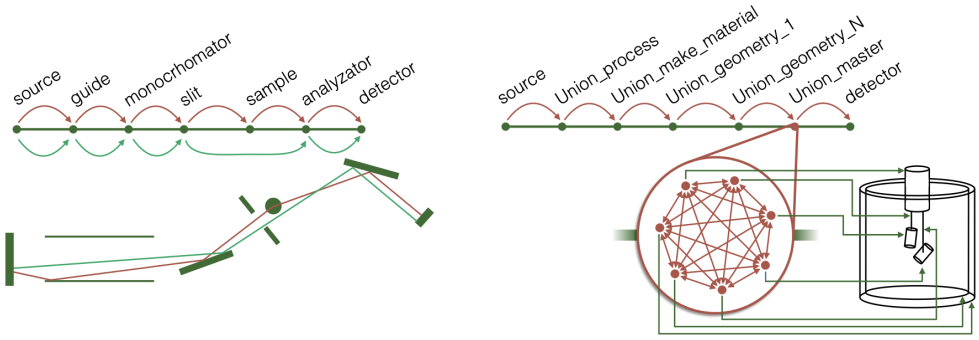
An ordinary McStas instrument is a linear sequence of self-contained components: a neutron ray enters a component, something happens to it (or not), and it moves on to the next component in the sequence. This is a good match for the intended beam path through an instrument, but it means that

- a single sample component must contain *all* the physics relevant to that sample – absorption, every scattering process, and any multiple scattering – as one monolithic block of code, and
- multiple scattering *between* physically separate parts of a sample environment (a sample inside a can, inside a cryostat, held by a sample stick, ...) is essentially impossible to capture, since each part would need to be a separate component and McStas components do not natively talk to each other.

This is a real limitation for complex sample environments, where multiple scattering between e.g. a powder sample and its aluminium container can be a significant background contribution.

The Union components solve this by splitting the sample simulation task into several cooperating component classes – *processes*, *materials*, *geometries* and a *master* – and performing the actual ray tracing for all of them together, inside the master component, with full native multiple scattering between every defined volume. Because physics (processes) and geometry are separated, adding a new scattering process is a comparatively small task, and existing processes can be freely combined and reused across different shapes.

It is entirely possible, and common, to mix ordinary McStas components with Union components in the same instrument file – the Union machinery only takes over at its `Union_master` components (section B.2).



Traditional McStas instrument

McStas instrument file using Union components

Figure B.1.: Left: the linear succession of components in a traditional McStas instrument file. Right: the same instrument using Union components – all ray tracing for the sample environment happens inside a single master component, which works around the normal McStas component sequence to achieve native multiple scattering between an arbitrary number of volumes.

B.1.1. Volumes, priority and overlap

The central concept of the Union components is the *volume*: a geometrical shape (box, cylinder, sphere, cone or triangle mesh) combined with a material definition (absorption plus zero or more scattering processes), placed in space using the ordinary `AT/ROTATED` mechanism. Volumes may be placed in any order in the instrument file, and, unlike ordinary McStas geometry, volumes are allowed to *overlap*.

Each volume is given a unique `priority` value. Wherever two or more volumes overlap in space, the material of the volume with the *highest* priority applies. This single rule makes it straightforward to build geometries that would otherwise require careful, error-prone manual splitting into non-overlapping pieces: a hollow cryostat wall is simply an aluminium cylinder with a vacuum cylinder of higher priority placed inside it; a window is a low-priority vacuum region cut into an otherwise higher-priority wall; a layered sample can with several different powders is built from concentric or adjacent shapes, each its own volume and material.

B.2. Component classes

A Union simulation is assembled from components of several distinct classes, always used in the same overall order:

1. **Process components** – define a scattering process (section B.2.2).

2. `Union_make_material` – collect processes and an absorption cross section into a named material (section B.2.3).

3. **Geometry components** – place a volume with a given material in the simulation (section B.2.4).

4. `Union_master` – performs the actual ray tracing for all volumes defined since the previous master (section B.2.5).

In addition, optional **logger** (section B.2.7) and **conditional** (section B.2.8) components can be placed to record detailed information about what happens inside a master component, and **surface** components (section B.2.6) can be attached to geometries to add reflection/refraction effects at their boundaries.

B.2.1. Loading the Union library

Union components need no setup or cleanup components. Each one loads the Union library itself, and the library keeps the internal bookkeeping lists (of processes, materials, geometries, loggers, ...) that the components communicate through, freeing them once the last Union component has finished.

An instrument that uses Union components must contain at least one `Union_master`; without one it fails to compile, with an error naming `Union_components_need_a_Union_master_in_t`. Union components placed after the last `Union_master` have no effect, and the simulation warns about each of them when it finishes.

`Union_init` and `Union_stop`, which older instruments place first and last among their Union components, are no longer needed. They are kept without effect, apart from a deprecation warning, so that such instruments still compile. For the same reason every Union component still accepts the string parameter `init`, which older instruments set to the name of their `Union_init` component; its value is ignored.

B.2.2. Scattering processes

A process component has no physical shape of its own. It describes a single physical scattering mechanism: a function giving the inverse penetration depth (macroscopic cross section) as a function of incoming wavevector, and a function describing what happens to a ray in a scattering event. Table B.1 lists the processes currently available; several are directly modelled on existing McStas sample components (e.g. `Incoherent_process` on `Incoherent`, `Powder_process` on `PowderN`, `Single_crystal_process` on `Single_crystal`), so users already familiar with those components will recognize most of the input parameters.

Every process has an `interact_fraction` setting parameter, in the range $[0, 1]$ or -1 to disable. When a material combines several processes, this can be used to artificially force a certain fraction of scattering events to that process, which is useful for improving statistics on a process that would otherwise rarely be sampled (an importance-sampling technique, entirely analogous to `p_interact` discussed below); the resulting ray weight is corrected automatically so the simulation remains unbiased. If left at -1 for every process in a material, the interact fractions default to the actual, physical scattering probabilities.

Component	Description
`Incoherent_process`	Isotropic, elastic incoherent scattering.
`Powder_process`	Bragg scattering from a powder (based on `PowderN`).
`Single_crystal_process`	Bragg scattering from a single crystal (based on `Single_crystal`); twinning is achieved by combining two or more instances with different orientations.
`Texture_process`	Single crystal scattering with an orientation distribution (texture) rather than a single fixed orientation.
`PhononSimple_process`	A single acoustic phonon branch.
`IncoherentPhonon_process`	Incoherent one-phonon scattering.
`AF_HB_1D_process`	Antiferromagnetic $S = 1/2$ 1D Heisenberg chain.
`Inhomogenous_incoherent_process`	Incoherent scattering with a spatially varying cross section.
`NCrystal_process`	Bragg and thermal-diffuse scattering via the external NCrystal library.
`Non_process`	A placeholder process with zero scattering probability.
`Template_process`	Documented template for writing a new process.

Table B.1.: Scattering processes currently available for the Union components.

A minimal example defining an incoherent process for Vanadium:

```
COMPONENT Vanadium_incoherent = Incoherent_process(
    sigma=5.08, unit_cell_volume=13.827, packing_factor=1)
AT (0,0,0) ABSOLUTE
```

A process component's position in the instrument file is irrelevant to the physics (it has no shape); its *name* is what matters, since materials refer to processes by name.

Contributed processes: MCViNE kernels

A substantial set of further processes, ported from the MCViNE sample-scattering kernels, is contributed in the `contrib` component category rather than `union` (table B.2); they follow the same Union process interface as the ones above and are used the same way. Each is paired with a standalone, non-Union component of the same name (without the `_process` suffix) sharing the same underlying kernel code.

A process does, however, use its rotation relative to ABSOLUTE to rotate its local frame. This can be used to create a small offset between two `Single_crystal_process` instances to describe a simple version of twinning. For that reason, always set the process rotation relative to ABSOLUTE.

Component	Description
`MCViNE_SQ_process`	Elastic isotropic scatterer with struc $S(\|Q\|)$ (expression or table).
`MCViNE_SQE_process`	Isotropic $S(Q,E)$ scatterer from an a pression or grid, with optional final-en ing.
`MCViNE_SvQ_process`	Elastic scatterer with a vector struc $S(Q_x,Q_y,Q_z)$ (expression or 3D grid
`MCViNE_E_Q_process`	Isotropic dispersion: $S(Q,E) = $ $E(Q))$ with analytic $E(Q)$, $S(Q)$.
`MCViNE_E_vQ_process`	Single-crystal dispersion: $S(Q,E) =$ $E(Q))$ with analytic $E(Q_x,Q_y,Q_z)$.
`MCViNE_Broadened_E_Q_process`	Isotropic dispersion $E(Q)$ broadened sian of Q-dependent width.
`MCViNE_LorentzianBroadened_E_Q_process`	Isotropic dispersion $E(Q)$ broade Lorentzian of Q-dependent width.
`MCViNE_ConstantEnergyTransfer_process`	Isotropic scatterer with a fixed energ
`MCViNE_ConstantQE_process`	Scatterer with fixed momentum trans energy transfer E.
`MCViNE_ConstantvQE_process`	Scatterer with a fixed momentum-tra Q, weighted by a Gaussian in E.
`MCViNE_Phonon_CoherentInelastic_PolyXtal_process`	Coherent one-phonon scattering from using a full phonon dispersion (energ larizations) on a grid.
`MCViNE_Phonon_CoherentInelastic_SingleXtal_process`	Coherent one-phonon scattering fro crystal, using a full phonon dispersio
`MCViNE_Phonon_IncoherentElastic_process`	Incoherent elastic scattering with a De factor.
`MCViNE_Phonon_IncoherentInelastic_process`	One-phonon incoherent inelastic scat a phonon density of states, optional f focusing.
`MCViNE_DGSSXRes_process`	Resolution kernel for direct-geome crystal spectrometers (aims at a pixe of-flight window).
`MCViNE_SANS2D_ongrid_process`	SANS from a 2D $S(Q_x,Q_y)$ map on a along z).

Table B.2.: Processes contributed from MCViNE, in the `contrib` component category.

B.2.3. Union_make_material

`Union_make_material` collects one or more previously defined processes into a named material, together with the absorption cross section (as the inverse penetration depth at

standard velocity, `my_absorption`, in m^{-1} at $v_0 = 2200$ m/s):

```
COMPONENT Al = Union_make_material(
    my_absorption=100*4*0.231/66.4,
    process_string="Al_incoherent,Al_powder")
AT (0,0,0) ABSOLUTE
```

If `process_string` is left unset, all processes defined since the previous `Union_make_material` are collected automatically; giving it explicitly is recommended, since it makes the resulting material self-documenting. Two material names are reserved and always available without being explicitly defined: `"Vacuum"` (no processes, no absorption) and `"Exit"` (behaves like vacuum, but see section B.2.4 below). Setting `absorber=1` instead of giving a `process_string` creates a purely absorbing material with no scattering processes at all.

For the common case of a material that is purely NCrystal, with no other processes to combine it with, `Union_NCrystal_material` is a shortcut that creates the underlying NCrystal process and the material in a single component:

```
COMPONENT Cu = Union_NCrystal_material(cfg="Cu_sg225.ncmat")
AT (0,0,0) ABSOLUTE
```

Its absorption is derived automatically from the same NCrystal configuration string, so, unlike `Union_make_material`, it takes no `my_absorption` parameter. Use `NCrystal_process` together with `Union_make_material` instead whenever NCrystal scattering needs to be combined with other processes in the same material.

B.2.4. Geometry components

A geometry component places a volume of a given shape and material in the simulation, using the material name from `Union_make_material` and the ordinary `AT`/`ROTATED` keywords for position and orientation. The currently available shapes are `Union_box`, `Union_cylinder`, `Union_sphere`, `Union_cone` and `Union_mesh` (an arbitrary triangle mesh, for shapes not covered by the other primitives). No ray tracing happens in a geometry component itself – it only registers the volume with the following `Union_master`.

Every geometry component shares a common set of setting parameters, summarized in table B.3, in addition to its own shape-specific dimensions (e.g. `radius`/`yheight` for `Union_cylinder`, `xwidth`/`yheight`/`zdepth` for `Union_box`).

```
COMPONENT cryostat_wall = Union_cylinder(
    radius=0.1, yheight=0.2, priority=10, material_string="Al")
AT (0,0,0) RELATIVE sample_position

COMPONENT cryostat_vacuum = Union_cylinder(
    radius=0.09, yheight=0.2, priority=11, material_string="Vacuum")
AT (0,0.01,0) RELATIVE sample_position
```

Here the vacuum cylinder (higher priority) hollows out the aluminium wall, and by displacing it 1 cm upward a solid bottom is left while the top stays open – a simple, direct application of the overlap-by-priority rule from section B.1.1.

Parameter	Description
material_string	Name of a material from Union_make_material, or "Vacuum"/"Exit".
priority	Unique priority value; the volume with the highest priority wins where volumes overlap (section B.1.1).
p_interact	Forces this probability [0-1] for a scattering event when a ray is inside the volume, regardless of path length (importance sampling; see note below).
visualize	Set to 0 to hide this volume in mcdisplay.
mask_string, mask_setting	Turn this volume into a *mask* restricting one or more other volumes (section B.2.4).
number_of_activations	Number of subsequent Union_master components this volume should be simulated by (section B.2.4).
surface_string parameters	Attach Union surface components (section B.2.6) to this volume's faces. The exact parameter names are shape-specific: Union_sphere has a single surface (plus cut_surface for internal cuts from being overlapped); Union_box and Union_cylinder instead expose one parameter per face (e.g. plus_x_surface/minus_x_surface/...for the box, curved_surface/top_surface/bottom_surface for the cylinder), plus a convenience all_face_surface to set every face at once. See the Component Manual for the exact list per shape.
target_index/ target_x,y,z, focus_aw/focus_ah, focus_xw/focus_xh, focus_r	Standard McStas focusing parameters, used by any process assigned to this volume's material that supports focusing.

Table B.3.: Setting parameters common to all Union geometry components.

p_interact and multiple scattering

Unlike p_interact in an ordinary McStas sample component, p_interact on a Union volume applies at *every* step of the multiple scattering loop, not just once. Setting it to 50%, for instance, gives a 25% chance of *two* scattering events in that volume. It

is therefore best kept well below 1; values close to 1 lead to very high probabilities of high-order multiple scattering and correspondingly large statistical weight corrections.

Masks

A mask volume restricts one or more other (*masked*) volumes: a ray only sees the masked volume's material where the masked volume and the mask volume(s) both cover the same space. This gives additional geometric freedom beyond what overlap-by-priority alone provides, and often reduces the number of volumes needed for an intricate shape. A masked volume can have several masks; `mask_setting` (`"All"` or `"Any"`, default `"All"`) controls whether every mask or just any one of them must cover a point for the masked volume to apply there. A mask volume is set via `mask_string` (a comma-separated list of the geometry names it masks) rather than `material_string`.

Exit volumes and number_of_activations

Normally, once a ray enters the region of space covered by a `Union_master`, it stays inside the Union ray-tracing loop until it escapes all defined volumes. Sometimes it is useful to break out of this loop early – for example to place an ordinary McStas monitor or component somewhere inside an otherwise Union-simulated geometry (a detector inside a detector tank, say). Assigning a volume the special material `"Exit"` achieves this: once a ray enters an exit volume, the current master stops and the *next* McStas component in the instrument sequence runs as normal; the ray does not automatically return to the Union simulation afterwards.

`number_of_activations` (default 1) controls how many subsequent `Union_master` components will simulate a given volume, which is useful when a static piece of geometry (a magnet, say) should be present across more than one master section of the instrument without redefining it.

B.2.5. Union_master

`Union_master` is where the actual ray tracing happens – it is the component that is placed into the ordinary linear McStas component sequence, and by default it simulates every volume defined since the previous master (or, for the first master, since the start of the instrument). If a volume is defined after the last `Union_master` in the file, it will never be simulated – remember to place a master after your geometry.

```
1 COMPONENT sample = Union_master()
2 AT (0,0,0) RELATIVE sample_position
```

Several of `Union_master`'s own setting parameters control the tagging system (section B.2.9) and numerical safety limits; see its entry in the Component Manual's Union chapter for the complete list. A GPU-enabled variant, `Union_master_GPU`, is available for use with the OpenACC build (see the User Manual's chapter on running McStas, section on GPU acceleration); the regular `Union_master` does not run on GPU.

The `SCATTER` keyword is used internally, in a non-standard way, to mark every point where a ray moves from one volume to another, so that `mcdisplay` can show the resulting path. A variable `number_of_scattering_events` and an array `scattered_flag` (indexed by volume) are made available for use in an `EXTEND` block on the master, if you want to set your own custom flags based on where and how many times a ray scattered:

```
COMPONENT sample = Union_master()
AT (0,0,0) RELATIVE sample_position
EXTEND
%{
  if (number_of_scattering_events > 0) scattered_flag = 1; else
      scattered_flag = 0;
%}
```

B.2.6. Surfaces

Surface components describe physics that happens *at the boundary* of a volume rather than throughout its bulk – reflection and refraction, as opposed to the scattering and absorption described by processes. `Mirror_surface` implements standard supermirror-style specular reflectivity (the same $R_0/Q_c/\alpha/m/W$ parametrization, or a reflectivity data file, as used elsewhere in McStas); a `Template_surface` is provided for writing new surface types. A surface component is attached to a geometry's `surface` parameter (its outer faces) or `cut_surface` parameter (faces created where the volume is cut by being overlapped by a higher-priority volume), by name, exactly like a process is attached to a material. Refraction and reflection handling are toggled globally on `Union_master` via its `enable_refraction`/`enable_reflection` parameters (both on by default).

B.2.7. Logger components

Because so much of the simulation happens inside one master component, it is not possible to insert an ordinary McStas monitor *inside* the Union ray tracing to see what is going on there. Logger components fill this gap: attached to one or more volumes (and optionally further restricted to specific process names via `target_process`), a logger records information about every scattering event that happens there and writes it out as an ordinary McStas monitor data file. Leaving `target_geometry` empty logs *all* volumes, including ones not yet defined at the point the logger appears in the instrument file. Table B.4 lists the currently available loggers; several further restrict what gets logged via an `order_total`/`order_volume`/`order_volume_process` parameter, so that only e.g. the first scattering event in a given volume is recorded.

A parallel set of *absorption loggers* (`Union_abs_logger_1D_space`, `_2D_space`, `_1D_time`, `_event`, `_nD`, and time-of-flight/ wavelength variants) record where and when absorption events happen instead of scattering events, which is often exactly what is needed for background studies of shielding and sample-environment components.

Component	Logs
`Union_logger_1D`	Time or $\|\mathbf{q}\|$ in a 1D histogram.
`Union_logger_2DQ`	(q_i, q_j) for a chosen pair of x, y, z.
`Union_logger_2D_kf`	As above, but the components of the final wavevector.
`Union_logger_2D_kf_time`	As above, with an additional time axis.
`Union_logger_2D_space`	Scattering position, 2D projection.
`Union_logger_2D_space_time`	As above, with an additional time axis.
`Union_logger_3D_space`	Scattering position in full 3D.

Table B.4.: Logger components currently available for the Union components.

B.2.8. Conditional components

A logger on its own shows *where* scattering happens; a conditional component answers the complementary question of what *ends up* contributing to a particular part of the final result. A conditional modifies one or more named loggers (via `target_loggers`) so that they only actually record an event if the ray's final state – once it finally leaves the Union simulation – satisfies some condition. `Union_conditional_standard` filters on final time and/or energy range and/or total scattering order; `Union_conditional_PSD` additionally requires the ray to intersect a virtual position-sensitive detector. Several conditionals can be applied to the same logger, in which case all of them must be satisfied. Setting `master_tagging=1` instead applies the conditional to the tagging system (section B.2.9) of the *next* master component, rather than to a specific logger.

Combining a position or scattering-vector logger with a conditional that filters, say, on a background region of the final detector signal, lets you directly inspect which volumes and processes actually contributed to that background – without having to re-run the simulation for each new question, as manual `EXTEND`-based tagging would require.

B.2.9. Tagging: ray histories

Independently of the logger system, `Union_master` can record a simple *history* for every ray: the ordered sequence of volumes entered and scattering processes undergone. All rays sharing the same sequence of volumes/processes count as the same history; histories are collected and, at the end of the simulation, written to a file (by default `Union_history.dat`) sorted by the total intensity they represent. An excerpt for a simple aluminium-can-with-powder setup:

```
1 V0 : Surrounding vacuum
2 V1 : powder_container          Material: Al         P0: Al_incoherent P1:
      Al_Powder
3 V2 : powder_inside_container   Material: Cu_powder  P0: Cu_incoherent  P1:
      Cu_powder
4 ------ Histories sorted after intensity ------
5 12221626   I=4.188166E-06   V0
```

```
 1882051   I=1.052731E-06   V0 -> V1 -> V2 -> V1 -> V0
 1517013   I=6.213315E-07   V0 -> V1 -> V0
  188661   I=8.043799E-08   V0 -> V1 -> V2 -> P0 -> V1 -> V0
```

`VX` refers to volume number X, `PX` to process number X *within whatever volume precedes it* in the string (so `P0` after `V1` and `P0` after `V2` are generally different physical processes, if V1 and V2 have different materials). This is invaluable for background investigations: to estimate how much intensity originates from scattering in a particular volume, search the history file for occurrences of `"VX -> P"`; to find only the part of that which reaches a particular detector, combine it with an exit volume placed there and search for histories ending in that volume's number. Tagging is controlled by `Union_master`'s `enable_tagging` and `history_limit` parameters (the latter caps memory use by limiting the number of distinct histories retained).

B.3. The propagation algorithm

This section gives a conceptual outline of how `Union_master` actually propagates rays; see the original thesis [Ber17] for the full derivation and pseudocode.

B.3.1. Basic Monte Carlo step

Within a volume, the distance to the next scattering position is drawn from the usual exponential distribution set by the total inverse penetration depth (sum over the volume's absorption and all its processes' scattering contributions). If this distance is less than the distance to the volume's boundary, a scattering event occurs there; the specific process is chosen by a weighted Monte Carlo choice between the processes active in the current material (or forced via `interact_fraction`, in which case the ray weight is corrected to remain unbiased). Otherwise, the ray is propagated to the boundary and the simulation continues in whichever volume lies on the other side.

B.3.2. Intersection and destination lists

Naively, determining "the other side" of a boundary would require testing every other defined volume for intersection and containment, for every propagation step – prohibitively expensive once more than a handful of volumes are involved. Instead, in the `initialize` section of each `Union_master`, the whole ensemble of volumes is analysed once to build, for every volume n:

- an *intersection list* I_n: the (usually small) set of other volumes that actually need an intersection test while the ray is inside volume n – volumes with lower priority, or volumes properly nested inside another candidate, are pruned out;
- a *destinations list* D_n: the set of volumes the ray could possibly end up in immediately after leaving volume n through its own boundary, used to resolve exactly which volume that is when several overlap there.

This turns an $O(N)$-per-step cost (checking every volume) into something close to $O(1)$ in typical geometries, and is what makes simulating sample environments built from dozens of volumes computationally practical. Figure B.2 sketches the resulting behaviour for a simple case.

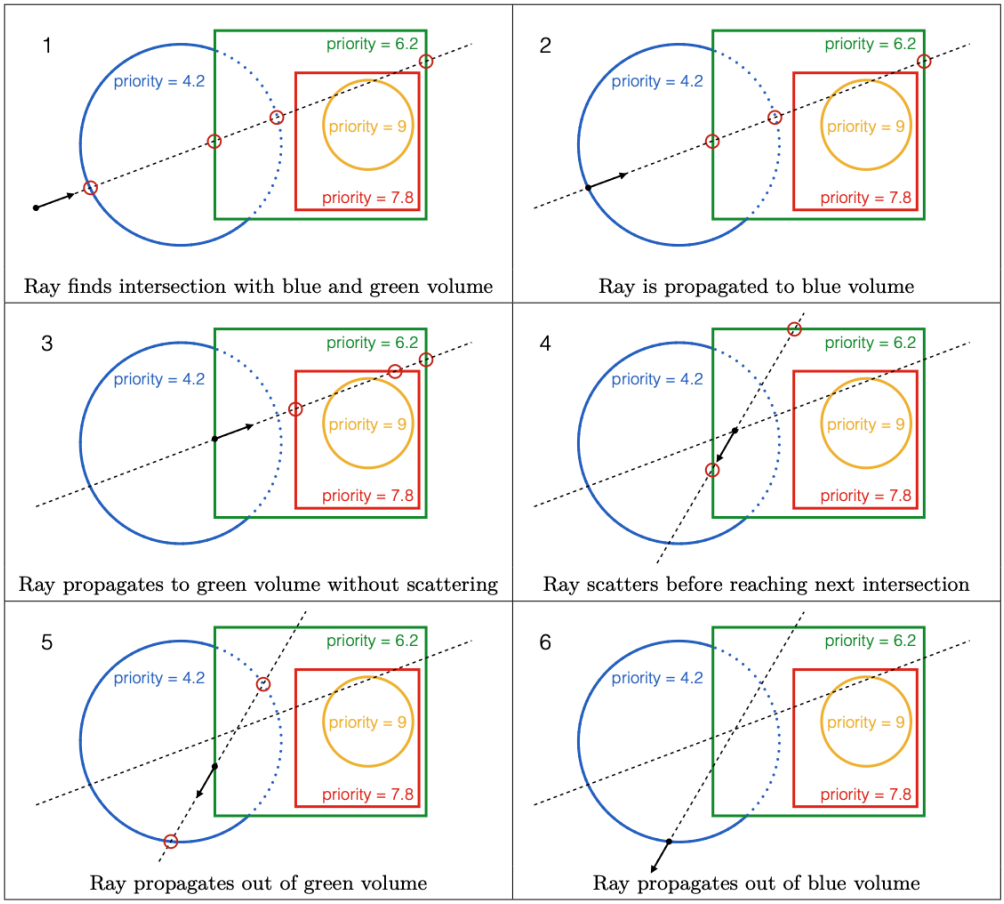


Figure B.2.: Illustration of the Union propagation algorithm for a single scattering event between two overlapping volumes. Only intersections actually relevant given the ray's current volume are calculated at each step (the orange volume, never reachable from the ray's actual path here, is never tested).

B.3.3. Known limitations

- **Coincident surfaces.** If two volume boundaries coincide exactly, the algorithm cannot reliably determine which volume a ray enters (the decision is essentially made by floating-point rounding). It is safe, and normal, to *overlap* volumes – but two volumes should never be placed so that a face merely touches without any overlap or gap; leaving even a micron-scale gap or overlap avoids the problem

entirely.

- **No gravity.** The underlying intersection algorithms are linear, so Union volumes do not support gravity; propagation inside a `Union_master` is always a straight line even if `--gravity` is enabled for the rest of the instrument.

- **GPU.** The default `Union_master` is CPU-only (`NOACC`); use `Union_master_GPU` for OpenACC builds (see the User Manual's chapter on running McStas, section on GPU acceleration).

B.4. A complete example

The following excerpt shows the pieces from this chapter assembled into a working instrument fragment: an aluminium sample can holding an iron sample, both with incoherent and powder scattering.

```
COMPONENT Al_inc = Incoherent_process(sigma=4*0.0082, unit_cell_volume
    =66.4)
AT (0,0,0) ABSOLUTE

COMPONENT Al_powder = Powder_process(reflections="Al.laz")
AT (0,0,0) ABSOLUTE

COMPONENT Al = Union_make_material(my_absorption=100*4*0.231/66.4,
    process_string="Al_inc,Al_powder")
AT (0,0,0) ABSOLUTE

COMPONENT Fe_inc = Incoherent_process(sigma=2*0.4, unit_cell_volume=24.04)
AT (0,0,0) ABSOLUTE

COMPONENT Fe_powder = Powder_process(reflections="Fe.laz")
AT (0,0,0) ABSOLUTE

COMPONENT Fe = Union_make_material(my_absorption=100*2*2.56/24.04,
    process_string="Fe_inc,Fe_powder")
AT (0,0,0) ABSOLUTE

/* ... source, guide etc. as normal ... */

COMPONENT sample_position = Arm()
AT (0,0,12.5) RELATIVE Guide

COMPONENT can = Union_cylinder(radius=0.006, yheight=0.04,
    priority=1, material_string="Al")
AT (0,0,0) RELATIVE sample_position

COMPONENT can_vacuum = Union_cylinder(radius=0.005, yheight=0.038,
    priority=2, material_string="Vacuum")
AT (0,0,0) RELATIVE sample_position

COMPONENT sample = Union_cylinder(radius=0.004, yheight=0.03,
```

```
35      priority=3, material_string="Fe")
36 AT (0,0,0) RELATIVE sample_position
37
38 COMPONENT master = Union_master()
39 AT (0,0,0) RELATIVE sample_position
40
41 COMPONENT Banana_monitor = Monitor_nD(radius=1, yheight=0.1,
42      options="banana, theta limits=[20,170], bins=200", filename="banana.dat
           ")
43 AT (0,0,0) RELATIVE sample_position
```

B.5. Further examples and worked tutorials

The example instrument library ships several dedicated Union categories: `Union_demos` (including `Manual_example`, essentially the worked example above), `Union_sample_environments`, `Union_validation` (comparisons against the traditional McStas components each process is based on) and `Tests_union` (unit-style tests of individual features). See the User Manual's chapter on instrument examples for how to browse these. A number of the individual component headers, documented in the Component Manual's Union chapter, contain further short usage examples.

For a more gradual, tutorial-style introduction covering geometry, masks, loggers, conditionals, exit volumes and tagging one step at a time, the McStasScript documentation's Union tutorial series is a useful complement to this chapter [Mcsb]. It is written using McStasScript's Python interface rather than the plain McStas instrument-file syntax used throughout this manual, but the underlying Union concepts – and the components themselves – are identical, so the explanations translate directly.

C. Libraries and constants

The McStas Library contains a number of built-in functions and conversion constants which are useful when constructing components. These are stored in the `share` directory of the `MCSTAS` library.

Within these functions, the 'Run-time' part is available for all component/instrument descriptions. The other parts are dynamic, that is they are not pre-loaded, but only imported once when a component requests it using the `%include` McStas keyword. For instance, within a component C code block, (usually SHARE or DECLARE):

```
1    %include "read_table-lib"
```

will include the 'read_table-lib.h' file, and the 'read_table-lib.c' (unless the `--no-runtime` option is used with `mcstas`). Similarly,

```
1    %include "read_table-lib.h"
```

will *only* include the 'read_table-lib.h'. The library embedding is done only once for all components (like the SHARE section). For an example of implementation, see **Res_monitor**.

In this Appendix, we present a short list of both each of the library contents and the run-time features.

C.1. Run-time calls and functions (`mcstas-r`)

Here we list a number of preprogrammed macros which may ease the task of writing component and instrument definitions.

C.1.1. Neutron propagation

Propagation routines perform all necessary operations to transport neutron rays from one point to an other. Except when using the special `ALLOW_BACKPROP;` call prior to executing any `PROP_*` propagation, the neutron rays which have negative propagation times are removed automatically.

- **ABSORB**. This macro issues an order to the overall McStas simulator to interrupt the simulation of the current neutron history and to start a new one.
- **PROP_Z0**. Propagates the neutron to the $z = 0$ plane, by adjusting (x, y, z) and t accordingly from knowledge of the neutron velocity (vx, vy, vz). If the propagation time is negative, the neutron ray is absorbed, except if a `ALLOW_BACKPROP;` preceeds it.

For components that are centered along the z-axis, use the `_intersect` functions to determine intersection time(s), and then a `PROP_DT` call.

- **PROP_DT**(dt). Propagates the neutron through the time interval dt, adjusting (x, y, z) and t accordingly from knowledge of the neutron velocity. This macro automatically calls PROP_GRAV_DT when the `--gravitation` option has been set for the whole simulation.

- **PROP_GRAV_DT**(dt, Ax, Ay, Az). Like **PROP_DT**, but it also includes gravity using the acceleration (Ax, Ay, Az). In addition to adjusting (x, y, z) and t, also (vx, vy, vz) is modified.

- **PROP_NO_G_DT**(dt). Propagates the neutron through the time interval dt using straight-line motion, regardless of the global gravitation setting. The negative-time and back-propagation rules are the same as for **PROP_DT**.

- **PROP_NO_G_X0**, **PROP_NO_G_Y0**, and **PROP_NO_G_Z0**. Propagate the neutron to the corresponding local coordinate plane using straight-line motion, regardless of the global gravitation setting.

- **ALLOW_BACKPROP**. Indicates that the next propagation routine will not remove the neutron ray, even if negative propagation times are found. Subsequent propagations are not affected.

- **SCATTER**. This macro is used to denote a scattering event inside a component. It should be used e.g to indicate that a component has interacted with the neutron ray (e.g. scattered or detected). This does not affect the simulation (see, however, **Beamstop**), and it is mainly used by the `MCDISPLAY` section and the `GROUP` modifier See also the SCATTERED variable (below).

C.1.2. Coordinate and component variable retrieval

- **MC_GETPAR**(*comp*, *outpar*). This may be used in e.g. the FINALLY section of an instrument definition to reference the output parameters of a component.

- **NAME_CURRENT_COMP** gives the name of the current component as a string.

- **POS_A_CURRENT_COMP** gives the absolute position of the current component. A component of the vector is referred to as POS_A_CURRENT_COMP.i where i is x, y or z.

- **ROT_A_CURRENT_COMP** and **ROT_R_CURRENT_COMP** give the orientation of the current component as rotation matrices (absolute orientation and the orientation relative to the previous component, respectively). A component of a rotation matrix is referred to as ROT_A_CURRENT_COMP[m][n], where m and n are 0, 1, or 2 standing for x, y and z coordinates respectively.

- **POS_A_COMP**(*comp*) gives the absolute position of the component with the name *comp*. Note that *comp* is not given as a string. A component of the vector is referred to as POS_A_COMP(*comp*).i where i is x, y or z.

- **ROT_A_COMP**(*comp*) and **ROT_R_COMP**(*comp*) give the orientation of the component *comp* as rotation matrices (absolute orientation and the orientation relative to its previous component, respectively). Note that *comp* is not given as a string. A component of a rotation matrice is referred to as ROT_A_COMP(*comp*)[m][n], where m and n are 0, 1, or 2.

- **INDEX_CURRENT_COMP** is the number (index) of the current component (starting from 1).

- **POS_A_COMP_INDEX**(*index*) is the absolute position of component *index*. POS_A_COMP_INDEX (INDEX_CURRENT_COMP) is the same as POS_A_CURRENT_COMP. You may use POS_A_COMP_INDEX (INDEX_CURRENT_COMP+1) to make, for instance, your component access the position of the next component (this is usefull for automatic targeting). A component of the vector is referred to as POS_A_COMP_INDEX(*index*).i where i is x, y or z.

- **POS_R_COMP_INDEX** works the same as above, but with relative coordinates.

- **STORE_NEUTRON**($index, x, y, z, vx, vy, vz, t, sx, sy, sz, p$) stores the current neutron state in the trace-history table, in local coordinate system. *index* is usually INDEX_CURRENT_COMP. This is automatically done when entering each component of an instrument.

- **RESTORE_NEUTRON**($index, x, y, z, vx, vy, vz, t, sx, sy, sz, p$) restores the neutron state to the one at the input of the component *index*. To ignore a component effect, use RESTORE_NEUTRON (INDEX_CURRENT_COMP, $x, y, z, vx, vy, vz, t, sx, sy, sz, p$) at the end of its TRACE section, or in its EXTEND section. These neutron states are in the local component coordinate systems.

- **SCATTERED** is a variable set to 0 when entering a component, which is incremented each time a SCATTER event occurs. This may be used in the `EXTEND` sections to determine whether the component interacted with the current neutron ray.

- **extend_list**(n, &*arr*, &*len*, *elemsize*). Given an array *arr* with *len* elements each of size *elemsize*, make sure that the array is big enough to hold at least n elements, by extending *arr* and *len* if necessary. Typically used when reading a list of numbers from a data file when the length of the file is not known in advance.

- **mcset_ncount**(n). Sets the number of neutron histories to simulate to n.

- **mcget_ncount**(). Returns the number of neutron histories to simulate (usually set by option `-n`).

- **mcget_run_num**(). Returns the number of neutron histories that have been simulated until now.

C.1.3. Coordinate transformations

- **coords_set**(x, y, z) returns a Coord structure (like POS_A_CURRENT_COMP) with x, y and z members.

- **coords_get**$(P, \&x, \&y, \&z)$ copies the x, y and z members of the Coord structure P into x, y, z variables.

- **coords_add**(a, b), **coords_sub**(a, b), **coords_neg**(a) enable to operate on coordinates, and return the resulting Coord structure.

- **rot_set_rotation**(*Rotation t*, ϕ_x, ϕ_y, ϕ_z) Get transformation matrix for rotation first ϕ_x around x axis, then ϕ_y around y, and last ϕ_z around z. t should be a 'Rotation' ([3][3] 'double' matrix).

- **rot_mul***(Rotation t1, Rotation t2, Rotation t3)* performs $t3 = t1.t2$.

- **rot_copy***(Rotation dest, Rotation src)* performs $dest = src$ for Rotation arrays.

- **rot_transpose***(Rotation src, Rotation dest)* performs $dest = src^t$.

- **rot_apply***(Rotation t, Coords a)* returns a Coord structure which is $t.a$

C.1.4. Mathematical routines

- **NORM**(x, y, z). Normalizes the vector (x, y, z) to have length 1 *.

- **scalar_prod**$(a_x, a_y, a_z, b_x, b_y, b_z)$. Returns the scalar product of the two vectors (a_x, a_y, a_z) and (b_x, b_y, b_z).

- **vec_prod**$(a_x,a_y,a_z, b_x,b_y,b_z, c_x,c_y,c_z)$. Sets (a_x, a_y, a_z) equal to the vector product $(b_x, b_y, b_z) \times (c_x, c_y, c_z)$ *.

- **rotate**$(x,y,z, v_x,v_y,v_z, \varphi, a_x,a_y,a_z)$. Set (x, y, z) to the result of rotating the vector (v_x, v_y, v_z) the angle φ (in radians) around the vector (a_x, a_y, a_z) *.

- **normal_vec**(n_x, n_y, n_z, x, y, z). Computes a unit vector (n_x, n_y, n_z) normal to the vector (x, y, z).*

- **solve_2nd_order**$(\&t_1, \&t_2\ A, B, C)$. Solves the 2nd order equation $At^2 + Bt + C = 0$ and returns the solutions into pointers $*t_1$ and $*t_2$. if t_2 is specified as `NULL`, only the smallest positive solution is returned in t_1.

(* The experienced c-programmer may be puzzled that these routines can return information without the use of *pass by reference*, the reason is that these calls are implemented as macros / `#define` wrapped functions.)

C.1.5. Output from detectors

Details about using these functions are given in the McStas User Manual.

- **DETECTOR_OUT_0D**(...). Used to output the results from a single detector. The name of the detector is output together with the simulated intensity and estimated statistical error. The output is produced in a format that can be read by McStas front-end programs.

- **DETECTOR_OUT_1D**(...). Used to output the results from a one-dimensional detector. Integrated intensities error etc. is also reported as for DETECTOR_OUT_0D.

- **DETECTOR_OUT_2D**(...). Used to output the results from a two-dimentional detector. Integrated intensities error etc. is also reported as for DETECTOR_OUT_0D.

- **mcinfo_simulation***(FILE *f, mcformat, char *pre, char *name)* is used to append the simulation parameters into file f (see for instance **Res_monitor**). Internal variable $mcformat$ should be used as specified. Please contact the authors for further information.

C.1.6. Ray-geometry intersections

- **inside_rectangle**(x, y, xw, yh). Return 1 if $-xw/2 \leq x \leq xw/2$ AND $-yh/2 \leq y \leq yh/2$. Else return 0.

- **box_intersect**(&t_1, &t_2, x, y, z, v_x, v_y, v_z, d_x, d_y, d_z). Calculates the (0, 1, or 2) intersections between the neutron path and a box of dimensions d_x, d_y, and d_z, centered at the origin for a neutron with the parameters (x, y, z, v_x, v_y, v_z). The times of intersection are returned in the variables t_1 and t_2, with $t_1 < t_2$. In the case of less than two intersections, t_1 (and possibly t_2) are set to zero. The function returns true if the neutron intersects the box, false otherwise.

- **cylinder_intersect**(&t_1, &t_2, x, y, z, v_x, v_y, v_z, r, h). Similar to **box_intersect**, but using a cylinder of height h and radius r, centered at the origin.

- **sphere_intersect**(&t_1, &t_2, x, y, z, v_x, v_y, v_z, r). Similar to **box_intersect**, but using a sphere of radius r.

C.1.7. Random numbers

- **rand01**(). Returns a random number distributed uniformly between 0 and 1.

- **randnorm**(). Returns a random number from a normal distribution centered around 0 and with $\sigma = 1$. The algorithm used to sample the normal distribution is explained in Ref. [Pre+86, ch.7].

- **randpm1**(). Returns a random number distributed uniformly between -1 and 1.

- **randtriangle**(). Returns a random number from a triangular distribution between -1 and 1.

- **randvec_target_circle**($\&v_x$, $\&v_y$, $\&v_z$, $\&d\Omega$, aim_x, aim_y, aim_z, r_f). Generates a random vector (v_x, v_y, v_z), of the same length as (aim_x, aim_y, aim_z), which is targeted at a *disk* centered at (aim_x, aim_y, aim_z) with radius r_f (in meters), and perpendicular to the *aim* vector.. All directions that intersect the circle are chosen with equal probability. The solid angle of the circle as seen from the position of the neutron is returned in $d\Omega$. This routine was previously called **randvec_target_sphere** (which still works).

- **randvec_target_rect_angular**($\&v_x, \&v_y, \&v_z, \&d\Omega, \mathrm{aim}_x, \mathrm{aim}_y, \mathrm{aim}_z, h, w, Rot$) does the same as randvec_target_circle but targetting at a rectangle with angular dimensions h and w (in **radians**, not in degrees as other angles). The rotation matrix *Rot* is the coordinate system orientation in the absolute frame, usually ROT_A_CURRENT_COMP.

- **randvec_target_rect**($\&v_x, \&v_y, \&v_z, \&d\Omega, \mathrm{aim}_x, \mathrm{aim}_y, \mathrm{aim}_z, height, width, Rot$) is the same as randvec_target_rect_angular but *height* and *width* dimensions are given in meters. This function is useful to e.g. target at a guide entry window or analyzer blade.

C.2. Reading a data file into a vector/matrix (Table input, `read_table-lib`)

The `read_table-lib` provides functionalities for reading text (and binary) data files. To use this library, add a `%include "read_table-lib"` in your component definition DECLARE or SHARE section. Tables are structures of type `t_Table` (see `read_table-lib.h` file for details):

```
/* t_Table structure (most important members) */
double *data;      /* Use Table_Index(Table, i j) to get element [i,j] */
long    rows;      /* number of rows */
long    columns;   /* number of columns */
char   *header;    /* the header with comments */
char   *filename;  /* file name or title */
double  min_x;     /* minimum value of 1st column/vector */
double  max_x;     /* maximum value of 1st column/vector */
```

Available functions to read *a single* vector/matrix are:

- **Table_Init**(&*Table*, *rows*, *columns*) returns an allocated Table structure. Use $rows = columns = 0$ not to allocate memory and return an empty table. Calls to Table_Init are *optional*, since initialization is being performed by other functions already.

- **Table_Read**(&*Table*, *filename*, *block*) reads numerical block number *block* (0 to concatenate all) data from *text* file *filename* into *Table*, which is as well initialized in the process. The block number changes when the numerical data changes its size, or a comment is encoutered (lines starting by '`# ; % /`'). If the data could not be read, then *Table.data* is NULL and $Table.rows = 0$. You may then try to read it using Table_Read_Offset_Binary. Return value is the number of elements read.

- **Table_Read_Offset**(&*Table*, *filename*, *block*, &*offset*, n_{rows}) does the same as Table_Read except that it starts at offset *offset* (0 means begining of file) and reads n_{rows} lines (0 for all). The *offset* is returned as the final offset reached after reading the n_{rows} lines.

- **Table_Read_Offset_Binary**(&*Table*, *filename*, *type*, *block*, &*offset*, n_{rows}, $n_{columns}$) does the same as Table_Read_Offset, but also specifies the *type* of the file (may be "float" or "double"), the number n_{rows} of rows to read, each of them having $n_{columns}$ elements. No text header should be present in the file.

- **Table_Rebin**(&*Table*) rebins all *Table* rows with increasing, evenly spaced first column (index 0), e.g. before using Table_Value. Linear interpolation is performed for all other columns. The number of bins for the rebinned table is determined from the smallest first column step.

- **Table_Info**(*Table*) print information about the table *Table*.

- **Table_Index**($Table, m, n$) reads the $Table[m][n]$ element.

- **Table_Value**($Table, x, n$) looks for the closest x value in the first column (index 0), and extracts in this row the n-th element (starting from 0). The first column is thus the 'x' axis for the data.

- **Table_Free**(&*Table*) free allocated memory blocks.

- **Table_Value2d**($Table, X, Y$) Uses 2D linear interpolation on a Table, from (X,Y) coordinates and returns the corresponding value.

Available functions to read *an array* of vectors/matrices in a *text* file are:

- **Table_Read_Array**(*File*, &n) read and split *file* into as many blocks as necessary and return a `t_Table` array. Each block contains a single vector/matrix. This only works for text files. The number of blocks is put into n.

- **Table_Free_Array**(&*Table*) free the *Table* array.

- **Table_Info_Array**(&*Table*) display information about all data blocks.

The format of text files is free. Lines starting by '`# ; % /`' characters are considered to be comments, and stored in *Table.header*. Data blocks are vectors and matrices. Block numbers are counted starting from 1, and changing when a comment is found, or the column number changes. For instance, the file 'MCSTAS/data/BeO.trm' (Transmission of a Berylium filter) looks like:

```
# BeO transmission, as measured on IN12
# Thickness: 0.05 [m]
# [ k(Angs-1) Transmission (0-1) ]
# wavevector multiply
1.0500  0.74441
1.0750  0.76727
1.1000  0.80680
...
```

Binary files should be of type "float" (i.e. REAL*32) and "double" (i.e. REAL*64), and should *not* contain text header lines. These files are platform dependent (little or big endian).

The *filename* is first searched into the current directory (and all user additional locations specified using the `-I` option, see the 'Running McStas' chapter in the User Manual), and if not found, in the `data` sub-directory of the `MCSTAS` library location. This way, you do not need to have local copies of the McStas Library Data files (see table 1.1).

A usage example for this library part may be:

```
t_Table Table;        // declare a t_Table structure
char file[]="BeO.trm"; // a file name
double x,y;

Table_Read(&Table, file, 1); // initialize and read the first numerical
    block
Table_Info(Table);           // display table informations
...
x = Table_Index(Table, 2,5); // read the 3rd row, 6th column element
                             // of the table. Indexes start at zero in C
                                .
y = Table_Value(Table, 1.45,1); // look for value 1.45 in 1st column (x
    axis)
                             // and extract 2nd column value of that row
Table_Free(&Table);          // free allocated memory for table
```

Additionally, if the block number (3rd) argument of **Table_Read** is 0, all blocks will be concatenated. The **Table_Value** function assumes that the 'x' axis is the first column (index 0). Other functions are used the same way with a few additional parameters, e.g. specifying an offset for reading files, or reading binary data.

This other example for text files shows how to read many data blocks:

```
t_Table *Table;       // declare a t_Table structure array
long     n;
double y;
```

```
4
5   Table = Table_Read_Array("file.dat", &n); // initialize and read the all
        numerical block
6   n = Table_Info_Array(Table);      // display informations for all blocks (
        also returns n)
7
8   y = Table_Index(Table[0], 2,5);   // read in 1st block the 3rd row, 6th
        column element
9                                     // ONLY use Table[i] with i < n !
10  Table_Free_Array(Table);          // free allocated memory for Table
```

You may look into, for instance, the source files for **Monochromator_curved** or **Virtual_input** for other implementation examples.

C.3. Monitor_nD Library

This library gathers a few functions used by a set of monitors e.g. Monitor_nD, Res_monitor, etc. It is also used by **Virtual_output**, a code-specific event-file writer that has moved to the **obsolete** component category (see the Component Manual's Obsolete chapter) and is superseded by **MCPL_output**; new instrument work should prefer MCPL. It may monitor any kind of data, create the data files, and may display many geometries (for `mcdisplay`). Refer to these components for implementation examples, and ask the authors for more details.

C.4. Adaptive importance sampling Library

This library is currently only used by the components **Source_adapt** and **Adapt_check**. It performs adaptive importance sampling of neutrons for simulation efficiency optimization. Refer to these components for implementation examples, and ask the authors for more details.

C.5. Vitess import/export Library

This library is used by the components **Vitess_input** and **Vitess_output**, as well as the `mcstas2vitess` utility. Both **Vitess_input** and **Vitess_output** have moved to the **obsolete** component category (see the Component Manual's Obsolete chapter) and are kept only for backward compatibility; new instrument work should prefer the portable, multi-code MCPL format (**MCPL_input**/**MCPL_output**, see the Misc chapter) over Vitess-specific event files. Refer to these components for implementation examples, and ask the authors for more details.

C.6. Constants for unit conversion etc.

The following predefined constants are useful for conversion between units

Name	Value	Conversion from	Conversion to
DEG2RAD	$2\pi/360$	Degrees	Radians
RAD2DEG	$360/(2\pi)$	Radians	Degrees
MIN2RAD	$2\pi/(360 \cdot 60)$	Minutes of arc	Radians
RAD2MIN	$(360 \cdot 60)/(2\pi)$	Radians	Minutes of arc
V2K	$10^{-10} \cdot m_{\mathrm{N}}/\hbar$	Velocity (m/s)	**k**-vector ($\mathrm{\AA}^{-1}$)
K2V	$10^{10} \cdot \hbar/m_{\mathrm{N}}$	**k**-vector ($\mathrm{\AA}^{-1}$)	Velocity (m/s)
VS2E	$m_{\mathrm{N}}/(2e)$	Velocity squared ($\mathrm{m^2\ s^{-2}}$)	Neutron energy (meV)
SE2V	$\sqrt{2e/m_{\mathrm{N}}}$	Square root of neutron energy ($\mathrm{meV^{1/2}}$)	Velocity (m/s)
FWHM2RMS	$1/\sqrt{8\log(2)}$	Full width half maximum	Root mean square (standard deviation)
RMS2FWHM	$\sqrt{8\log(2)}$	Root mean square (standard deviation)	Full width half maximum
MNEUTRON	$1.67492 \cdot 10^{-27}$ kg	Neutron mass, m_{n}	
HBAR	$1.05459 \cdot 10^{-34}$ Js	Planck constant, $\hbar$	
PI	3.14159265...	π	
FLT_MAX	3.40282347E+38F	a big float value	

D. The McStas terminology

This is a short explanation of phrases and terms which have a specific meaning within McStas. We have tried to keep the list as short as possible with the risk that the reader may occasionally miss an explanation. In this case, you are more than welcome to contact the McStas core team.

- **Arm** A generic McStas component which defines a frame of reference for other components.
- **Component** One unit (*e.g.* optical element) in a neutron spectrometer. These are considered as Types of elements to be instantiated in an Instrument description.
- **Component instance** A named Component (of a given Type) inserted in an Instrument description.
- **Definition parameter** An input parameter for a component. For example the radius of a sample component or the divergence of a collimator.
- **Input parameter** For a component, either a definition parameter or a setting parameter. These parameters are supplied by the user to define the characteristics of the particular instance of the component definition. For an instrument, a parameter that can be changed at simulation run-time.
- **Instrument** An assembly of McStas components defining a neutron spectrometer.
- **Kernel** The McStas meta-language definition and the associated compiler `mcstas`.
- **McStas** Monte Carlo Simulation of Triple Axis Spectrometers (the name of this package). Pronunciation ranges from *mex-tas*, to *mac-stas* and *m-c-stas*.
- **Output parameter** An output parameter for a component. For example the counts in a monitor. An output parameter may be accessed from the instrument in which the component is used using `MC_GETPAR`.
- **Run-time** C code, contained in the files `mcstas-r.c` and `mcstas-r.h` included in the McStas distribution, that declare functions and variables used by the generated simulations.
- **Setting parameter** Similar to a definition parameter, but with the restriction that the value of the parameter must be a number.

Bibliography

[Mcsa] See `https://www.mcstas.org` (cit. on pp. 18, 25).

[Git] See `https://github.com/mccode-dev/McCode/issues` (cit. on pp. 18, 26).

[Nmi] See `http://neutron-eu.net/en` (cit. on p. 18).

[Mcn] See `http://mcnsi.risoe.dk` (cit. on p. 18).

[Ts2] See `http://ts-2.isis.rl.ac.uk` (cit. on p. 18).

[Ess] See `https://www.ess.eu` (cit. on p. 18).

[Sin] See `http://sine2020.eu` (cit. on p. 18).

[Ics] ICSD, Inorganic Crystal Structure Database. See `http://icsd.ill.fr` (cit. on pp. 22, 24, 166).

[Cry] Crystallographica, Oxford Cryosystem, 1998. See `http://www.crystallographica.com` (cit. on p. 166).

[Ful] Fullprof powder refinement. See `http://www-llb.cea.fr/fullweb/fp2k/fp2k.htm` (cit. on p. 166).

[Mcsb] See `https://panosc-vinyl.github.io/McStasScript/index.html` (cit. on p. 767).

[Bac75] G.E. Bacon. *Neutron Diffraction*. Oxford University Press, 1975 (cit. on pp. 133, 150, 173).

[Ber17] Mads Bertelsen. "Software for simulation and design of neutron scattering instrumentation". PhD thesis. University of Copenhagen, 2017 (cit. on pp. 246, 754, 764).

[BJL17] Mads Bertelsen, Henrik Jacobsen, and Kim Lefmann. "Expanding the McStas sample simulation logic with McStas Union components". In: *Review of Scientific Instruments* 88 (2017), p. 085106. DOI: `10.1063/1.4994697` (cit. on pp. 246, 754).

[Bla83] Y. Blanc. In: *ILL Report* 83BL21G (1983) (cit. on p. 93).

[Cla+98] K. N. Clausen et al. "The RITA spectrometer at Risø - design considerations and recent results". In: *Physica B* 241-243 (1998), pp. 50–55 (cit. on p. 75).

[Cop74] J.R.D Copley. In: *Comput. Phys. Commun.* 7 (1974), p. 289 (cit. on pp. 182, 185–187).

[Cop93] J.R.D. Copley. In: *J. Neut. Research* 1 (1993), p. 21 (cit. on p. 27).

[Cus03] L. D. Cussen. In: *J. Appl. Cryst.* 36 (2003), p. 1204 (cit. on p. 216).

[DL03] A.-J. Dianoux and G. Lander. *ILL Neutron Data Booklet*. OCP Science, 2003 (cit. on pp. 22, 24, 168, 185).

[Ege67] P.A. Egelstaff. *An introduction to the liquid state*. Academic Press, London, 1967 (cit. on p. 183).

[FMW72] Bischoff FG, Yeater ML, and Moore WE. In: *Nuclear Science and Engineering* 48 (1972), p. 266 (cit. on pp. 181, 184, 185, 187).

[Far+02] E. Farhi et al. In: *Appl. Phys.* A 74 (2002), S1471 (cit. on p. 27).

[FBS06] H.E. Fischer, A.C. Barnes, and P.S. Salmon. In: *Rep. Prog. Phys.* 69 (2006), p. 233 (cit. on p. 183).

[GRR92] Grimmett, G. R., and Stirzakerand D. R. *Probability and Random Processes, 2nd Edition*. Clarendon Press, Oxford, 1992 (cit. on p. 27).

[Jam80] F. James. In: *Rep. Prog. Phys.* 43 (1980), p. 1145 (cit. on pp. 27, 31).

[Johrc] M.W. Johnson. In: *Harwell report* AERE - R 7682 (March 1974) (cit. on pp. 181, 185–187).

[LN99] K. Lefmann and K. Nielsen. "McStas, a general software package for neutron ray-tracing simulations". In: *Neutron News* 10 (1999), pp. 20–23. DOI: 10.1080/10448639908233684 (cit. on p. 18).

[Lie05] K. Lieutenant. In: *J. Phys.: Condens. Matter* 17 (2005), S167 (cit. on p. 27).

[Lin+15] Jiao Y. Y. Lin et al. "MCViNE – An object oriented Monte Carlo neutron ray tracing simulation package". In: *Nuclear Instruments and Methods in Physics Research Section A* 810 (2015). arXiv:1504.02776, https://arxiv.org/abs/1504.02776, pp. 86–99 (cit. on p. 639).

[Lov84] S.W. Lovesey. *Theory of neutron scattering from condensed matter*. Oxford Clarendon Press, 1984 (cit. on pp. 739, 742, 743, 745).

[MPC77] D.F.R. Mildner, C.A. Pellizari, and J.M. Carpenter. In: *Acta. Cryst. A* 33 (1977), p. 954 (cit. on p. 185).

[Pre+86] W. H. Press et al. *Numerical Recipes in C*. Cambridge University Press, 1986 (cit. on p. 773).

[Pre+02] W.H. Press et al. *Numerical Recipes (2nd Edition)*. Cambridge University Press, 2002 (cit. on p. 93).

[Sch+04] C. Schanzer et al. In: *Nucl. Instr. Meth.* A 529 (2004), p. 63 (cit. on p. 27).

[Sea75] V.F. Sears. In: *Adv. Phys.* 24 (1975), p. 1 (cit. on p. 184).

[SD01] P. A. Seeger and L. L. Daemen. "Numerical Solution of Bloch's Equation for Neutron Spin Precession". In: *Nucl. Instr. Meth.* 457 (2001), p. 338 (cit. on p. 740).

[Squ78] G.L. Squires. *Thermal Neutron Scattering*. Cambridge University Press, 1978 (cit. on pp. 150, 151, 159, 172, 182, 188).

[Wil+14] Peter Kjær Willendrup et al. "McStas: Past, present and future". In: *Journal of Neutron Research* 17.1 (2014), pp. 35–43. ISSN: 1023-8166. DOI: 10.3233/JNR-130004 (cit. on p. 18).

[Wil+05] Peter Willendrup et al. *User and Programmers Guide to the Neutron Ray-Tracing Package McStas, Version 1.9*. Risoe Report, 2005 (cit. on pp. 18, 25).

[Wil88] W. Gavin Williams. *Polarized Neutrons*. Oxford Clarendon Press, 1988 (cit. on pp. 739, 741, 745).

[ZLa04] G. Zsigmond, K. Lieutenant, and S. Manoshin et al. In: *Nucl. Instr. Meth.* A 529 (2004), p. 218 (cit. on p. 27).

Index

`%include` (McStas keyword), 768

`ABSORB` (C macro, mcstas-r.h), *768*
adapt_tree-lib (library), **776**
Adaptive sampling, 30
`ALLOW_BACKPROP` (C macro, mcstas-r.h), *769*
Arm, *778*

`box_intersect` (C function, mccode-r.c), *772*
Bugs, 25, 79, 80, 169

C functions, 768
Coherent and incoherent isotropic scatterer, 180
Component, *778*
  instance, *778*
Concentric components, 148, 180
Constants, **776**
Contrib, **487**
Conversion
  of units, 776
Coordinate
  retrieval functions, 769
  system, 21
`coords_add` (C function, mccode-r.c), *771*
`coords_get` (C function, mccode-r.c), *771*
`coords_set` (C function, mccode-r.c), *771*
`cylinder_intersect` (C function, mccode-r.c), *772*

Data files, 21
Definition parameter, *778*
`DEG2RAD` (constant), **777**
`DETECTOR_OUT_0D` (C macro, mccode-r.h), *772*
`DETECTOR_OUT_1D` (C macro, mccode-r.h), *772*
`DETECTOR_OUT_2D` (C macro, mccode-r.h), *772*
Diffraction, 148, 158, 168
Direction focusing, 30

Environment variable
  BROWSER, 25
  MCSTAS, 23
Environment variables
  `MCSTAS`, 768, 775
Error estimate, 28
`EXTEND` (McStas keyword), 769

`FLT_MAX` (constant), **777**
Focusing
  importance sampling, 30
`FWHM2RMS` (constant), **777**

`GROUP` (McStas keyword), 769

`HBAR` (constant), **777**

Importance sampling
  direction focusing, 30
Incoherent elastic scattering, 138, 158
Incoherent inelastic scattering, 141
`INDEX_CURRENT_COMP` (C macro, mccode-r.h), *770*
Inelastic scattering, 171, 180
Input parameter, *778*
`inside_rectangle` (C function, mcstas-r.c), *772*
Instrument, *778*

`K2V` (constant), **777**
Kernel, *778*
Keyword
  EXTEND, 25, 33, 228
  OUTPUT PARAMETERS, 229

Library, **768**
  adapt_tree-lib, **776**
  Components
    contrib, 487
    data, 22, 24
    misc, 237
    monitors, 205
    obsolete, 694
    optics, 59, 85, 105
    samples, 132

sasmodels, 327
sources, 33
union, 246
components
data, 775
share, 768
mccode-r, **768**
mcstas-r, **768**
monitor_nd-lib, **776**
pol-lib, 746
read_table-lib, **773**
read_table-lib (Read_Table), 21
Run-time
MC_GETPAR, 229
run-time, **768**
vitess-lib, **776**

`MC_GETPAR` (C macro, mccode-r.h), *769*
mccode-r (library), **768**
`MCDISPLAY` (McStas keyword), 769
MCNP, 58
`MCSTAS` (environment variable), 768, 775
McStas
name, 778
pronunciation, 778
mcstas-r (library), **768**
`mcstas2vitess` (McStas tool), 776
MCViNE, 639, 668
`MIN2RAD` (constant), **777**
`MNEUTRON` (constant), **777**
monitor_nd-lib (library), **776**
Monitors, **205**
Adaptive importance sampling monitor, 51
Banana shape, 227
Beam analyzer, 245
Capture flux, 224
Custom monitoring (user variables, Monitor_nD), 227
Divergence monitor, 214
Divergence/position monitor, 216
Energy monitor, 210
Neutron parameter correlations, PreMonitor_nD, 230
Number of neutron bounces in a guide, 229
Position sensitive detector (PSD), 213
Position sensitive monitor recording mean energy, 227
The All-in-One monitor (Monitor_nD), 222
Time-of-flight monitor, 207
TOF2E monitor, 209
Wavelength monitor, 211
Monte Carlo method, 27
accuracy, 31
adaptive sampling, 30
direction focusing, 30
stratified sampling, 30
Multiple scattering, 158, 180

`NAME_CURRENT_COMP` (C macro, mccode-r.h), *769*
Neutron polarisation, *see* Polarisation
`NORM` (C macro, mccode-r.h), *771*
`normal_vec` (C function, mccode-r.c), *771*

Obsolete, **694**
Optics, **59**, **73**, **74**, **85**, **105**
Beam stop, 62
Bender (non polarizing), 80
Curved guides (polygonal model), 81
Disc chopper, 86
Fermi Chopper, 80, **90**, 97
Guide with channels (straight, non focusing), 79
Guide with channels and gravitation handling (straight), 80
Linear collimator, 68
Mirror plane, 74
Monochromator, 122
Monochromator, curved, 127
Monochromator, polarizing, 129
Monochromator, thick, 131
phase space transformer, 131
Point in space (Arm, Optical bench), 59
Radial collimator, 71
Slit, 61
Straight guide, 76
Velocity selector, 101, 104
Optimization, 47, 51, 53, 57
Output parameter, *778*

`PI` (constant), **777**
Polarisation, **105**, **739**
field functions, 748
monitors, 232
Monochromator_pol, 129

precession algorithm, **746**
SimpleNumMagnetPrecession, 747
POS_A_COMP (C macro, mccode-r.h), *770*
POS_A_COMP_INDEX (C macro, mccode-r.h), *770*
POS_A_CURRENT_COMP (C macro, mccode-r.h), *769*
POS_R_COMP_INDEX (C macro, mccode-r.h), *770*
Preprocessor macros, 768
PROP_DT (C macro, mcstas-r.h), *769*
PROP_GRAV_DT (C macro, mcstas-r.h), *769*
PROP_NO_G_DT (C macro, mcstas-r.h), *769*
PROP_NO_G_X0 (C macro, mcstas-r.h), *769*
PROP_NO_G_Y0 (C macro, mcstas-r.h), *769*
PROP_NO_G_Z0 (C macro, mcstas-r.h), *769*
PROP_Z0 (C macro, mcstas-r.h), *768*

RAD2DEG (constant), **777**
RAD2MIN (constant), **777**
rand01 (C function, mccode-r.c), *772*
randnorm (C function, mccode-r.c), *773*
randpm1 (C function, mccode-r.c), *773*
randtriangle (C function, mccode-r.c), *773*
randvec_target_circle (C function, mccode-r.c), *773*
randvec_target_rect (C function, mccode-r.c), *773*
randvec_target_rect_angular (C function, mccode-r.c), *773*
read_table-lib (library), **773**
Removed neutron events, 21, 189, 769
RESTORE_NEUTRON (C macro, mcstas-r.h), *770*
RMS2FWHM (constant), **777**
ROT_A_COMP (C macro, mccode-r.h), *770*
ROT_A_CURRENT_COMP (C macro, mccode-r.h), *769*
rot_apply (C function, mccode-r.c), *771*
rot_copy (C function, mccode-r.c), *771*
rot_mul (C function, mccode-r.c), *771*
ROT_R_COMP (C macro, mccode-r.h), *770*
ROT_R_CURRENT_COMP (C macro, mccode-r.h), *769*
rot_set_rotation (C function, mccode-r.c), *771*
rot_transpose (C function, mccode-r.c), *771*
rotate (C macro, mccode-r.h), *771*
Run-time, *778*

Sample environments, 148, 180, 193
Samples, **132**
Coherent and incoherent isotropic scatterer, 180
Dilute colloid medium, 168
Incoherent inelastic scatterer, 141
Incoherent isotropic scatterer (Vanadium), 138
Phonon scattering, 171
Powder, multiple diffraction line, 148
Single crystal diffraction, 158
Sampling, 30
SANS, 135
SasView, **327**
scalar_prod (C function, mccode-r.c), *771*
SCATTER (C macro, mcstas-r.h), *769*, 770
SCATTERED (C macro, mccode-r.h), *770*
SE2V (constant), **777**
Setting parameter, *778*
SHARE (McStas keyword), 768
Simulation progress bar, 245
Small angle scattering, 168
solve_2nd_order (C function, mccode-r.c), *771*
Sources, **33**
Adaptive importance sampling monitor, 51
Adaptive source, 47
Continuous source with a Maxwellian spectrum, 39
Continuous source with specified divergence, 38
General continuous source, 43
ISIS pulsed moderators, 488
Optimization location, *see* Sources/Optimizer
Optimizer, 53
Simple continuous source, 36
Time of flight pulsed moderator, 45
sphere_intersect (C function, mccode-r.c), *772*
Spin-echo, 752
Statistics
uncertainty, 28
STORE_NEUTRON (C macro, mcstas-r.h), *770*
Stratified sampling, 30
Symbols, 21

`Table_Free` (C function, read_table-lib.c), *774*
`Table_Free_Array` (C function, read_table-lib.c), *774*
`Table_Index` (C function, read_table-lib.c), *774*
`Table_Info` (C function, read_table-lib.c), *774*
`Table_Info_Array` (C function, read_table-lib.c), *775*
`Table_Init` (C function, read_table-lib.c), *774*
`Table_Read` (C function, read_table-lib.c), *774*
`Table_Read_Array` (C function, read_table-lib.c), *774*
`Table_Read_Offset` (C function, read_table-lib.c), *774*
`Table_Read_Offset_Binary` (C function, read_table-lib.c), *774*
`Table_Rebin` (C function, read_table-lib.c), *774*
`Table_Value` (C function, read_table-lib.c), *774*
`Table_Value2d` (C function, read_table-lib.c), *774*
Tools
  mcdoc, 25
Tripoli, 58
Union, **246**, **754–767**
  algorithm, 764
  conditionals, 763
  exit volume, 761
  geometry, 759
  loggers, 762
  masks, 761
  master, 761
  materials, 758
  MCViNE, 668, 757
  processes, 756
  surfaces, 762
  tagging, 763
  Union_init, 756
  Union_stop, 756
Units
  constants and conversions, **776**

`V2K` (constant), **777**
Variance, 28
Variance reduction, 30
`vec_prod` (C function, mccode-r.c), *771*
Virtual sources, 31
Vitess, 58
vitess-lib (library), **776**
`VS2E` (constant), **777**

Weight, **28**
  statistical uncertainty, 28
  transformation, 29