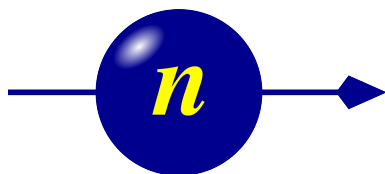




Physics Physics Department,  
Technical University of Denmark  
2800 Kongens Lyngby, Denmark

# Component Manual for the Neutron Ray-Tracing Package McStas, version 3.7



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

May, 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 3.7, released May, 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, Den-  
mark

Present address: University of Lund, Lund, Sweden  
Klaus Lieutenant <k.lieutenant@fz-juelich.de>  
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, Den-  
mark  
Presently associated with: MySQL AB, Sweden

ISBN 978-87-550-3680-2

ISSN 0106-2840

Physics Department · DTU · 2026

# Contents

<b>Preface and acknowledgements</b>	<b>8</b>
<b>1. About the component library</b>	<b>10</b>
1.1. Authorship . . . . .	10
1.2. Symbols for neutron scattering and simulation . . . . .	10
1.3. Component coordinate system . . . . .	11
1.4. About data files . . . . .	11
1.5. Component source code . . . . .	12
1.6. Documentation . . . . .	12
1.7. Component validation . . . . .	15
1.8. Disclaimer, bugs . . . . .	15
<b>2. Monte Carlo Techniques and simulation strategy</b>	<b>16</b>
2.1. Neutron spectrometer simulations . . . . .	16
2.1.1. Monte Carlo ray tracing simulations . . . . .	16
2.2. The neutron weight . . . . .	17
2.2.1. Statistical errors of non-integer counts . . . . .	17
2.3. Weight factor transformations during a Monte Carlo choice . . . . .	18
2.3.1. Direction focusing . . . . .	19
2.4. Adaptive and Stratified sampling . . . . .	19
2.5. Accuracy of Monte Carlo simulations . . . . .	20
<b>3. Source components</b>	<b>22</b>
3.0.1. Neutron flux . . . . .	22
3.1. The <code>Source_simple</code> McStas Component . . . . .	24
3.2. The <code>Source_div</code> McStas Component . . . . .	25
3.3. The <code>Source_Maxwell_3</code> McStas Component . . . . .	26
3.4. The <code>Source_gen</code> McStas Component . . . . .	28
3.5. The <code>Moderator</code> McStas Component . . . . .	32
3.6. The <code>Source_adapt</code> McStas Component . . . . .	34
3.7. The <code>Adapt_check</code> McStas Component . . . . .	35
3.8. The <code>Source_Optimizer</code> McStas Component . . . . .	37
3.9. The <code>Monitor_Optimizer</code> McStas Component . . . . .	39
3.10. Other sources components: contributed pulsed sources, virtual sources (event files) . . . . .	41
<b>4. Beam optical components: Arms, slits, collimators, and filters</b>	<b>42</b>
4.1. The <code>Arm</code> McStas Component . . . . .	42

4.2.	The <code>Slit</code> McStas Component . . . . .	43
4.3.	The <code>Beamstop</code> McStas Component . . . . .	44
4.4.	<code>Filter_gen</code> : A general filter using a transmission table . . . . .	45
4.5.	The <code>Collimator_linear</code> McStas Component . . . . .	46
4.6.	The <code>Collimator_radial</code> McStas Component . . . . .	47
<b>5.</b>	<b>Reflecting optical components: mirrors, and guides</b>	<b>49</b>
5.1.	The <code>Guide</code> McStas Component . . . . .	49
<b>6.</b>	<b>Moving optical components: Choppers and velocity selectors</b>	<b>51</b>
6.1.	The <code>DiskChopper</code> McStas Component . . . . .	51
6.2.	The <code>FermiChopper</code> McStas Component . . . . .	54
6.3.	The <code>Vitess_ChopperFermi</code> McStas Component . . . . .	55
6.4.	The <code>V_selector</code> McStas Component . . . . .	59
6.5.	The <code>Selector</code> McStas Component . . . . .	60
<b>7.</b>	<b>Monochromators</b>	<b>62</b>
7.1.	The <code>Monochromator_flat</code> McStas Component . . . . .	62
7.2.	The <code>Monochromator_curved</code> McStas Component . . . . .	63
7.3.	<code>Single_crystal</code> : Thick single crystal monochromator plate with multiple scattering . . . . .	66
7.4.	Phase space transformer - moving monochromator . . . . .	66
<b>8.</b>	<b>Samples</b>	<b>67</b>
8.0.1.	Neutron scattering notation . . . . .	67
8.0.2.	Weight transformation in samples; focusing . . . . .	68
8.0.3.	Future development of sample components . . . . .	70
8.1.	The <code>Incoherent</code> McStas Component . . . . .	70
8.2.	The <code>Tunneling_sample</code> McStas Component . . . . .	74
8.3.	The <code>PowderN</code> McStas Component . . . . .	76
8.4.	The <code>Single_crystal</code> McStas Component . . . . .	82
8.5.	The <code>Sans_spheres</code> McStas Component . . . . .	89
8.6.	The <code>Phonon_simple</code> McStas Component . . . . .	91
8.7.	The <code>Isotropic_Sqw</code> McStas Component . . . . .	93
<b>9.</b>	<b>Monitors and detectors</b>	<b>100</b>
9.1.	The <code>TOF_monitor</code> McStas Component . . . . .	101
9.2.	The <code>TOF2E_monitor</code> McStas Component . . . . .	102
9.3.	The <code>E_monitor</code> McStas Component . . . . .	103
9.4.	The <code>L_monitor</code> McStas Component . . . . .	104
9.5.	The <code>PSD_monitor</code> McStas Component . . . . .	105
9.6.	The <code>Divergence_monitor</code> McStas Component . . . . .	106
9.7.	The <code>DivPos_monitor</code> McStas Component . . . . .	107
9.8.	The <code>Monitor_nD</code> McStas Component . . . . .	110

<b>10. Special-purpose components</b>	<b>116</b>
10.1. Virtual_output: Saving the first part of a split simulation . . . . .	117
10.2. Virtual_input: Starting the second part of a split simulation . . . . .	117
10.3. Res_sample: A sample-like component for resolution calculation . . . . .	118
10.4. TOF_Res_sample: A sample-like component for TOF resolution calculation	118
10.5. Res_monitor: The monitor for resolution calculation . . . . .	119
10.6. Progress_bar: Simulation progress and automatic saving . . . . .	121
10.7. Beam_spy: A beam analyzer . . . . .	121
<b>A. Polarization in McStas</b>	<b>122</b>
A.1. Introduction . . . . .	122
A.2. The Polarization Vector . . . . .	122
A.2.1. Example: Magnetic fields . . . . .	124
A.3. Polarized Neutron Scattering . . . . .	125
A.3.1. Example: Nuclear scattering . . . . .	126
A.3.2. Example: Polarizing Monochromator and Guides . . . . .	127
A.4. New McStas Components . . . . .	129
A.4.1. Polarizers . . . . .	129
A.4.2. Detectors . . . . .	130
A.4.3. Magnetic fields . . . . .	130
A.4.4. Samples . . . . .	131
A.5. Tests With New Components . . . . .	131
<b>B. Libraries and constants</b>	<b>133</b>
B.1. Run-time calls and functions ( <b>mcstas-r</b> ) . . . . .	133
B.1.1. Neutron propagation . . . . .	133
B.1.2. Coordinate and component variable retrieval . . . . .	134
B.1.3. Coordinate transformations . . . . .	136
B.1.4. Mathematical routines . . . . .	136
B.1.5. Output from detectors . . . . .	137
B.1.6. Ray-geometry intersections . . . . .	137
B.1.7. Random numbers . . . . .	137
B.2. Reading a data file into a vector/matrix (Table input, <b>read_table-lib</b> ) .	138
B.3. Monitor_nD Library . . . . .	141
B.4. Adaptive importance sampling Library . . . . .	141
B.5. Vitess import/export Library . . . . .	141
B.6. Constants for unit conversion etc. . . . .	141
<b>C. The McStas terminology</b>	<b>143</b>
<b>Bibliography</b>	<b>144</b>
<b>Index and keywords</b>	<b>145</b>

# 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 3.7. The initial release in October 1998 of version 1.0 was presented in Ref. [LN99] and further developed through 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 [Mcs]. 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 altogether.

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; Mcna]. 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-learning” 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 B.
- An explanation of the McStas terminology in Appendix C.

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  $\mathbf{k} = m_n \mathbf{v} / \hbar$ , where  $m_n$  is the neutron mass.  $\mathbf{k}$  is usually given in  $\text{\AA}^{-1}$ , 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  $\mathbf{s}$  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 Appendix 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 B.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 ??) and Isotropic\_Sqw components (see ??). Whenever using any  $S(q, \omega)$  data for the Isotropic\_Sqw component, we kindly request to cite the corresponding reference.

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

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] [ $\omega$ ] [ $S(q, \omega)$ ]

Table 1.1.: Data files of the McStas library.

## 1.5. Component source code

Source code for all components may be found in the **MCSTAS** library subdirectory of the McStas installation; the default is `/usr/local/lib/mcstas/` on Unix-like systems and `C:\mcstas\lib` on Windows systems, but it may be changed using the **MCSTAS** environment variable.

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 --text <name>
2 mcdoc <file .comp>
```

The first line will search for all components matching the *name*, and display their help section as text. For instance, **mcdoc .laz** will list all available Lazy data files, whereas **mcdoc --text 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.

MCSTAS/data File name	$\sigma_{coh}$ [barns]	$\sigma_{inc}$ [barns]	$\sigma_{abs}$ [barns]	$T_m$ [K]	$c$ [m/s]	Note
Ag.laz	4.407	0.58	<b>63.3</b>	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	<b>98.65</b>	1337.4	<b>1740</b>	
B4C.laz	19.71	6.801	<b>3068</b>	2718		
Ba.laz	3.23	0.15	29.0	1000	<b>1620</b>	
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	<b>1790</b>	
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	<b>2520</b>	594.2	2310	
Cr.laz	1.660	1.83	3.05	2180	5940	
Cs.laz	3.69	0.21	29.0	301.6	<b>1090</b>	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	<b>49700</b>	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	<b>372.3</b>	234.32	<b>1407</b>	
I2.laz	7.0	0.62	12.3	386.85		
In.laz	2.08	0.54	<b>193.8</b>	429.75	<b>1215</b>	
K.laz	.69	0.27	2.1	336.53	<b>2000</b>	
LiF.laz	4.46	0.921	<b>70.51</b>	1140		
Li.laz	0.454	0.92	<b>70.5</b>	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	<b>1260</b>	
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	<b>1300</b>	
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	<b>818</b>	
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  $\sigma_{abs}$  materials are highlighted. Files are given in LAZY format, but may exist as well in Crystallographica .lau format as well.

MCSTAS/data File name	$\sigma_{coh}$ [barns]	$\sigma_{inc}$ [barns]	$\sigma_{abs}$ [barns]	$T_m$ [K]	$c$ [m/s]	Note
Cs_liq_tot.sqw	3.69	0.21	29.0	301.6	<b>1090</b>	liquid cesium. densteiner et al, Phys. Rev. A (1992) 5709 and (1992) 3574.
D2_liq_21_tot.sqw	7.64	0	0.00052	21		liquid deuterium. Measured.
D2_pow_21_tot.sqw	7.64	0	0.00052	12		et al, PHYSICAL REVIEW B (2009) 064301
D2O_liq_290_coh and D_liq_290_inc	15.4	4.1	0.0012	290		powder deuterium. Measured.
D2O_liq_290_tot	15.4	4.1	0.0012	290		et al, PHYSICAL REVIEW B (2009) 064301
Ge_liq_coh.sqw and Ge_liq_inc.sqw	8.42	0.18	2.2	1211.4	5400	liquid heavy water. Classical MD. Farhi et al, J. Nucl. Sci. Tech. (2015)
H2O_liq_290_coh and H2O_liq_290_inc	7.75	161	0.665	290	1530	liquid heavy water. Measured. E. Farhi et al, J. Nucl. Sci. Tech. (2015)
H2O_liq_290_tot	7.75	161	0.665	290	1530	liquid germanium. Ab-initio MD. Hugouvieux Farhi E, John MR, et al, Phys. Rev. B 75 (2007) 104201
He4_liq_coh.sqw	1.34	0	0.00747	2	<b>240</b>	liquid water. Classical MD. E. Farhi et al, J. Nucl. Sci. Tech. (2015)
He4_liq_1_coh.sqw	1.34	0	0.00747	1	<b>240</b>	liquid water. Measured. E. Farhi et al, J. Nucl. Sci. Tech. (2015)
Ne_liq_tot.sqw	2.62	0.008	0.039	24.56	<b>591</b>	liquid helium. Measured (constant width). Donnelly et al, Low Temp. Phys. 44 (1981) 471
Rb_liq_coh.sqw and Rb_liq_inc.sqw	6.32	0.5	0.38	312.46	<b>1300</b>	liquid helium. Measured. Andersen et al, Phys Cond Mat (1994) 821

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

## 1.7. Component validation

Some components were checked for release 1.9: the Fermi choppers, the velocity selectors, 2 of the guide components and Source\_gen. The results are summarized in a talk available online (<http://www.ill.fr/tas/mcstas/doc/ValMcStas.pdf>).

Velocity selector and Fermi chopper were treated as black boxes and the resulting line shapes cross-checked against analytical functions for some cases. The component 'Selector' showed no dependence on the distance between guide and selector axis. This is corrected at the moment. Apart from that the component yielded correct results. That was different with the Fermi chopper components. The component 'Chopper\_Fermi', which has been part of the McStas distribution for a long time, gave wrong results and was removed from the package. The new 'Vitesse\_ChopperFermi' (transferred from the VITESS package) showed mainly correct behaviour. Little bugs were corrected after the first tests. At the moment, there is only the problem left that it underestimates the influence of a shadowing cylinder. With the contributed 'FermiChopper' component, there were also minor problems, which are all corrected in the meantime.

For the guides, several trajectories through different kinds of guides (straight, convergent, divergent) were calculated analytically and positions, directions and losses of reflections compared to the values calculated in the components. This was done for 'Guide' and 'Guide\_gravity'; in the latter case calculations were performed with and without gravity. Additionally a cross-check against the VITESS guide module was performed. Waviness, chamfers and channels were not checked. After correction of a bug in 'Guide\_gravity', both components worked perfectly (within the conditions tested).

'Source\_gen' was cross-checked against the VITESS source module for the case of 3 Maxwellians describing the moderator characteristic and typical sizes the guide and its distance to the moderator. It showed the same line shape as a function of wavelength and divergence and the same absolute values.

## 1.8. Disclaimer, bugs

We would like to emphasize that the usage of both the McStas software, as well as its components are the responsibility 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.

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

In the case where the  $u_i$  values are regularly sampled, we come to the well known midpoint integration rule. In the case where 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  $\pi_j$ . The resulting weight factor for the neutron ray after passage of the  $n$  components in the instrument becomes the product of all contributions

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

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

$$I = \sum_i p_i = N\bar{p}, \quad (2.3)$$

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,  $p_i \approx \bar{p}$ , and that we observe a large number of neutrons,  $N \geq 10$ . Then  $N$  almost follows a normal distribution with the uncertainty  $\sigma(N) = \sqrt{N}$ <sup>1</sup>. Hence, the statistical uncertainty of the observed intensity becomes

$$\sigma(I) = \sqrt{N}\bar{p} = I/\sqrt{N}, \quad (2.4)$$

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

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

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

$$\sigma^2(I) = N(\bar{p}^2 + \sigma^2(p_i)). \quad (2.6)$$

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

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

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

$$\sigma^2(I) \approx \sum_i p_i^2. \quad (2.8)$$

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

---

<sup>1</sup>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:  $f_{\text{MC}} = 1$ . 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  $f_{\text{MC}} = P, \pi_j = 1$ . To generalize, weight factor transformations are given by the master equation

$$f_{\text{MC}}\pi_j = P. \quad (2.9)$$

This probability rule is general, and holds also if, e.g., it is decided to transmit only half of the rays ( $f_{\text{MC}} = 0.5$ ). 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  $4\pi$  solid angle, and all interesting directions are known to be contained within a certain solid angle interval  $\Delta\Omega$ , only these solid angles are used for the Monte Carlo choice of scattering direction. This implies  $f_{\text{MC}}(\Delta\Omega) = 1$ . 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

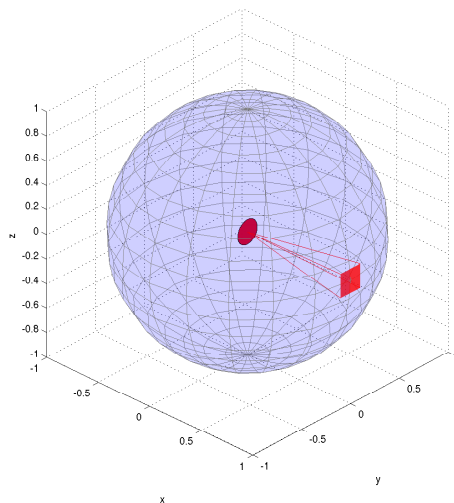


Figure 2.1.: Illustration of the effect of direction focusing in McStas. Weights of neutrons emitted into a certain solid angle are scaled down by the full unit sphere area.

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*randpml(); /* 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,  $\psi$ , at an instrument is defined as intensity per area perpendicular to the beam direction.

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

$$I_{\text{total}} = \Phi A \Delta\Omega \Delta\lambda, \quad (3.1)$$

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).

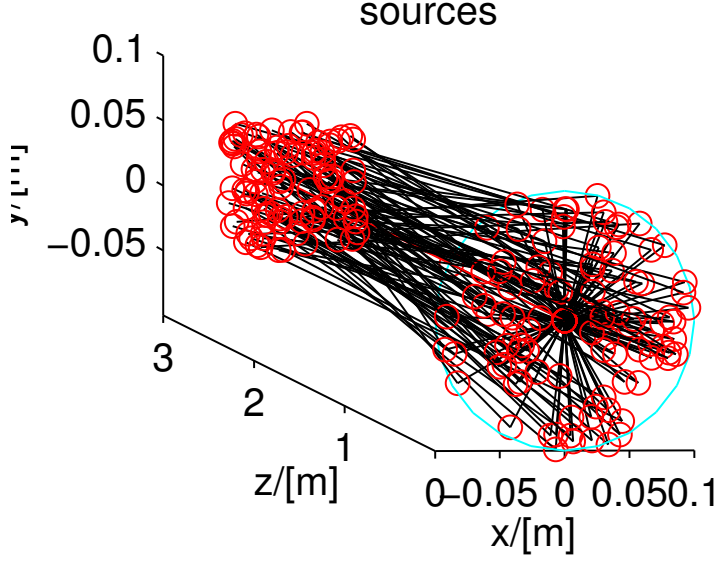


Figure 3.1.: A circular source component (at  $z=0$ ) emitting neutron events randomly, either from a model, or from a data file.

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  $N_{\text{sim}}$  denotes the number of neutron histories to simulate, the initial neutron weight  $p_0$  must be set to

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

where the source flux is now given a  $\lambda$ -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 **Mod-erator** for time-of-flight sources.

Optimizers can dramatically improve the statistics, but may occasionally give wrong results, due to misled 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

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

#### Description

```
1 The routine is a circular neutron source, which aims at a square target
2 centered at the beam (in order to improve MC-acceptance rate). The angular
3 divergence is then given by the dimensions of the target.
4 The neutron energy is uniformly distributed between lambda0-dlambda and
5 lambda0+dlambda or between E0-dE and E0+dE.
6 The flux unit is specified in n/cm2/s/st/energy unit (meV or Angs).
7
8 This component replaces Source_flat, Source_flat_lambda,
9 Source_flux and Source_flux_lambda.
10
11 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



Name	Unit	Description	Default
lambda0	AA	Mean wavelength of neutrons.	0
dlambda	AA	Wavelength half spread of neutrons.	0
flux	1/(s*cm**2*flux energy unit, Angs or meV if flux=0, the source emits 1 in 4*PI whole space.		1
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`.

## 3.2. The Source\_div McStas Component

Neutron source with Gaussian or uniform divergence

### Identification

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

### Description

```

1 The routine is a rectangular neutron source, which has a gaussian or
  uniform
2 divergent output in the forward direction.
3 The neutron energy is distributed between lambda0-dlambda and
4 lambda0+dlambda or between E0-dE and E0+dE. The flux unit is specified
5 in n/cm2/s/st/energy unit (meV or Angs).
6 In the case of uniform distribution (gauss=0), angles are uniformly
  distributed
7 between -focus_aw and +focus_aw as well as -focus_ah and +focus_ah.
8 For Gaussian distribution (gauss=1), 'focus_aw' and 'focus_ah' define the
9 FWHM of a Gaussian distribution. Energy/wavelength distribution is also
10 Gaussian.
11
12 Example: Source_div(xwidth=0.1, yheight=0.1, focus_aw=2, focus_ah=2, E0=14,
  dE=2, gauss=0)

```

```

13
14 %VALIDATION
15 Feb 2005: tested by Kim Lefmann      (o.k.)
16 Apr 2005: energy distribution used in external tests of Fermi choppers (o.k
   .)
17 Jun 2005: wavelength distribution used in external tests of velocity
   selectors (o.k.)
18 Validated by: K. Lieutenant
19
20 %BUGS
21 distribution is uniform in (hor. and vert.) angle (relative to moderator
   normal) ,
22 therefore not suited for large angles

```

## Input parameters

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

Name	Unit	Description	Default
<b>xwidth</b>	m	Width of source	
<b>yheight</b>	m	Height of source	
<b>focus_aw</b>	deg	FWHM (Gaussian) or maximal (uni- form) horz. width divergence	
<b>focus_ah</b>	deg	FWHM (Gaussian) or maximal (uni- form) vert. height divergence	
E0	meV	Mean energy of neutrons.	0.0
dE	meV	Energy half spread of neutrons.	0.0
lambda0	Ang	Mean wavelength of neutrons (only rele- vant for E0=0)	0.0
dlambda	Ang	Wavelength half spread of neutrons.	0.0
gauss	0 1	Criterion: 0: uniform, 1: Gaussian dis- tributions	0
flux	1/(s cm 2 st en- ergy_unit)	flux per energy unit, Angs or meV	1

## Links

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

## 3.3. The Source\_Maxwell1\_3 McStas Component

Source with up to three Maxwellian distributions

## Identification

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

## Description

```
1 A parametrised continuous source for modelling a (cubic) source
2 with (up to) 3 Maxwellian distributions.
3 The source produces a continuous spectrum.
4 The sampling of the neutrons is uniform in wavelength.
5
6 Units of flux: neutrons/cm^2/second/ster
7 (McStas units are in general neutrons/second)
8
9 Example: PSI cold source T1=150.42 K / 2.51 AA      I1 = 3.67 E11
10 T2=38.74 K / 4.95 AA      I2 = 3.64 E11
11 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
<b>Lmin</b>	AA	Lower edge of lambda distribution	
<b>Lmax</b>	AA	Upper edge of lambda distribution	
<b>dist</b>	m	Distance from source to focusing rectangle; at (0,0,dist)	
<b>focus_xw</b>	m	Width of focusing rectangle	
<b>focus_yh</b>	m	Height of focusing rectangle	
<b>T1</b>	K	1st temperature of thermal distribution	
T2	K	2nd temperature of thermal distribution	300
T3	K	3rd temperature of - - -	300
<b>I1</b>	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

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
lambda0	AA	Mean wavelength of neutrons.	0
dlambda	AA	Wavelength spread of neutrons.	0

## Links

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

## 3.4. The Source\_gen McStas Component

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

### Identification

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

### Description

```

1 This routine is a neutron source (rectangular or circular), which aims at
2 a square target centered at the beam (in order to improve MC-acceptance
3 rate). The angular divergence is then given by the dimensions of the
4 target. However, it may be directly set using the 'focus-aw' and 'focus_ah'
5 parameters.
6
7 The neutron energy/wavelength is distributed uniformly in wavelength
8 between
9 Emin=E0-dE and Emax=E0+dE or Lmin=lambda0-dlambda and Lmax=lambda0+dlambda.
10 The I1 may be either arbitrary (I1=0), or specified in neutrons per
11 steradian
12 per square cm per Angstrom per s. A Maxwellian spectra may be selected if
13 you
14 give the source temperatures (up to 3).
15
16 Finally, a file with the flux as a
17 function of the wavelength [lambda(AA) flux(n/s/cm^2/st/AA)] may be used
18 with the 'flux_file' parameter. Format is 2 columns free text.
19
20 Additional distributions for the horizontal and vertical phase spaces
21 distributions (position-divergence) may be specified with the

```

```

19 'xdiv_file' and 'ydiv_file' parameters. Format is free text, requiring
20 a comment line '# xylimits: pos_min pos_max div_min div_max' to set
21 the axis of the distribution matrix. All these files may be generated using
22 standard monitors (better in McStas/PGPLOT format), e.g.:
23 Monitor_nD(options="auto lambda per cm2")
24 Monitor_nD(options="x hdiv, all auto")
25 Monitor_nD(options="y vdiv, all auto")
26
27 The source shape is defined by its radius, or can alternatively be squared
28 if you specify non-zero yheight and xwidth parameters.
29 The beam is divergence uniform,.
30 The source may have a thickness, which will broaden the default zero time
31 distribution.
32
33 Usage example:
34 Source_gen(radius=0.1,lambda0=2.36,dlambda=0.16,T1=20,I1=1e13,focus_xw
    =0.01,focus_yh=0.01)
35 Source_gen(yheight=0.1,xwidth=0.1,Emin=1,Emax=3,I1=1e13,verbose=1,focus_xw
    =0.01,focus_yh=0.01)
36 EXTEND
37 %{
38 t = rand0max(1e-3); // set time from 0 to 1 ms for TOF instruments.
39 %}
40
41 <b>Some neutron facility parameters:</b>
42 PSI cold source      T1=296.2,I1=8.5E11, T2=40.68,I2=5.2E11
43 ILL VCS cold source T1=216.8,I1=1.24e+13,T2=33.9,I2=1.02e+13
44 (H1, 58 MW) T3=16.7 ,I3=3.0423e+12
45 ILL HCS cold source T1=413.5,I1=10.22e12,T2=145.8,I2=3.44e13
46 (H5, 58 MW) T3=40.1 ,I3=2.78e13
47 ILL Thermal tube    T1=683.7,I1=0.5874e+13,T2=257.7,I2=2.5099e+13
48 (H12, 58 MW) T3=16.7 ,I3=1.0343e+12
49 ILL Hot source      T1=1695, I1=1.74e13,T2=708, I2=3.9e12 (58MW)
50 HZB cold source     T1=43.7 ,I1=1.4e12, T2=137.2,I2=2.08e12,radius=.155 (10
    MW)
51 HZB bi-spectral     T1=43.7, I1=1.4e12, T2=137.2,I2=2.08e12,T3=293.0,I3
    =1.77e12
52 HZB thermal tube    T1=293.0,I1=2.64e12 (10MW)
53 FRM2 cold,20MW      T1=35.0, I1=9.38e12,T2=547.5,I2=2.23e12,T3=195.4,I3
    =1.26e13
54 FRM2 thermal,20MW   T1=285.6,I1=3.06e13,T2=300.0,I2=1.68e12,T3=429.9,I3
    =6.77e12
55 LLB cold,14MW       T1=220, I1=2.09e12,T2=60, I2=3.83e12,T3=20, I3
    =1.04e12
56 TRIGA thermal 1MW   T1=300, I1=3.5e11 (scale by thermal power in MW)
57
58 %VALIDATION
59 Feb 2005: output cross-checked for 3 Maxwellians against VITESS source
60 I(lambda), I(hor_div), I(vert_div) identical in shape and absolute values
61 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 <b>&lt;b&gt;either&lt;/b&gt;</b> $[1/(s \cdot cm^2 \cdot st)]$ <b>&lt;b&gt;or&lt;/b&gt;</b> $[1/(s \cdot cm^2 \cdot st \cdot AA)]$ (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
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	AA	Mean wavelength of neutrons.	0
dlambda	AA	Wavelength spread of neutrons, half width	0
I1	$1/(cm^2 \cdot sr)$	Source flux per solid angle, area and Angstrom if I1=0, the source emits 1 in $4 \cdot \pi$ whole space.	1

Name	Unit	Description	Default
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 Aangstrom. 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	AA	Minimum wavelength of neutrons	0
Lmax	AA	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`.
- P. Ageron, Nucl. Inst. Meth. A 284 (1989) 197

## 3.5. The Moderator McStas Component

A simple pulsed source for time-of-flight.

### Identification

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

### Description

```
1 Produces a simple time-of-flight spectrum, with a flat energy distribution
2
3 Example: Moderator(radius = 0.0707, dist = 9.035, focus_xw = 0.021,
    focus_yh = 0.021, Emin = 10, Emax = 15, Ec = 9.0, t0 = 37.15, gamma =
    39.1)
```

### Input parameters

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

Name	Unit	Description	Default
radius	m	Radius of source	0.07
<b>Emin</b>	meV	Lower edge of energy distribution	
<b>Emax</b>	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 <sup>2</sup> st meV)	flux	1



## Links

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

## 3.6. The Source\_adapt McStas Component

Neutron source with adaptive importance sampling

### Identification

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

### Description

```
1 Rectangular source with flat energy or wavelength distribution that
2 uses adaptive importance sampling to improve simulation efficiency.
3 Works together with the Adapt_check component.
4
5 The source divides the three-dimensional phase space of (energy ,
6 horizontal position , horizontal divergence) into a number of
7 rectangular bins. The probability for selecting neutrons from each
8 bin is adjusted so that neutrons that reach the Adapt_check
9 component with high weights are emitted more frequently than those
10 with low weights. The adjustment is made so as to attempt to make
11 the weights at the Adapt_check components equal.
12
13 Focusing is achieved by only emitting neutrons towards a rectangle
14 perpendicular to and placed at a certain distance along the Z axis.
15 Focusing is only approximate (for simplicity); neutrons are also
16 emitted to pass slightly above and below the focusing rectangle ,
17 more so for wider focusing.
18
19 In order to prevent false learning , a parameter beta sets a
20 fraction of the neutrons that are emitted uniformly , without regard
21 to the adaptive distribution . The parameter alpha sets an initial
22 fraction of neutrons that are emitted with low weights; this is
23 done to prevent early neutrons with rare initial parameters but
24 high weight to ruin the statistics before the component adapts its
25 distribution to the problem at hand. Good general-purpose values
26 for these parameters are alpha = beta = 0.25.
27
28 %VALIDATION
29 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
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	AA	Mean wavelength of neutrons (if energy not specified)	0
dlambda	AA	Wavelength spread half width	0
flux		(1/(cm <sup>2</sup> AA 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 ( $0 < \alpha \leq 1$ )	0.25
beta	1	Aggressiveness of adaptive algorithm ( $0 < \beta \leq 1$ )	0.25

## Links

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

## 3.7. The Adapt\_check McStas Component

Optimization specifier for the Source\_adapt component.

## Identification

- **Site:**
- **Author:** Kristian Nielsen

- **Origin:** Risoe
- **Date:** 1999

## Description

```

1 This components works together with the Source_adapt component, and
2 is used to define the criteria for selecting which neutrons are
3 considered "good" in the adaptive algorithm. The name of the
4 associated Source_adapt component in the instrument definition is
5 given as parameter. The component is special in that its position
6 does not matter; all neutrons that have not been absorbed prior to
7 the component are considered "good".
8
9 Example: Adapt_check(source_comp="MySource")

```

## Input parameters

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

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

## Links

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

## 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

- **Site:**
- **Author:** <mailto:farhi@ill.fr> Emmanuel Farhi
- **Origin:** <http://www.ill.fr> ILL (France)
- **Date:** 17 Sept 1999

### Description

```
1 Principle: The optimizer first (step 1) computes neutron state parameter
2 limits passing in the Source_Optimizer, and then (step 2) records a
   Reference
3 source as well as the state (at Source_Optimizer position) of neutrons
4 reaching Monitor. The optimized source is defined as a fraction of the
5 Reference source plus the distribution of 'good' neutrons reaching the
6 Monitor. The optimization then starts (step 3), and focuses new neutrons on
7 the Monitor_Optimizer. In fact it changes 'bad' neutrons into 'good' ones
8 (that reach the Monitor), acting on their position, spin and divergence or
9 velocity. The overall Monitor flux is kept during process. The energy and
10 polarisation distributions are kept during optimization as far as possible
11 during optimisation. The optimization method considers that all neutron
12 parameters — (x,y), (vx,vy,vz) or (vx/v2,vy/v2,v2), (sx,sy,sz) or
13 (sx/s2,sy/s2,s2) — are independent.
14
15 Options: The optimized source can be computed regularly ('continuous'
16 option) or only once ('not continuous'). The time spent in steps 1 and 2
   can
17 be reduced for a shorter optimization ('auto'). The neutrons passing
   during
18 steps 1 and 2 can be smoothed for a better neutron weight distribution
19 ('smooth' option).
20
21 Source_optimizer can be placed at any position where you want to act on the
22 flux, for instance just after the source.
23 Monitor_Optimizer should be placed at position(s) to optimize.
24 I prefer to put one just before the sample.
25
26 Default parameters bins, step, and keep are 10, 10% and 10% respectively.
27 The option string can be empty ("), which stands for default configuration
28 that works fine in usual cases:
29
30 options="continuous optimization, auto mode, smooth, SetXY+SetDivV+SetDivS"
31
32 <b>Possible options are</b>
```

```

33 continuous      for continuous source optimization (default).
34 verbose         displays optimization process (debug purpose).
35 auto           uses the shortest possible 'step 1' and 'step 2' and sets '
    step' value as required (default).
36 smooth         remove possible spikes generated in steps 1 and 2 (default
    is smooth).
37 inactivate      to inactivate the Optimizer.
38 no or not      revert next option
39 bins=[value=10] set the Number of cells for sampling neutron states
40 step=[value=10] Optimizer step in % of simulation.
41 keep=[value=10] Percentage of initial source distribution that is kept
42 file=[name]      Filename where to save optimized source distributions (no
    file is generated if not given. Default ext. is .src)
43 SetXY           Keywords to indicate what may be changed during
44 SetV            optimisation. Default is position, divergence and spin
45 SetS            direction ("SetXY+SetDivV+SetdivS"). Choosing the speed
46 SetDivV         or spin optimization (SetV or SetS) may modify the energy
47 SetDivS         or polarisation distribution (norm of V and S) as the three
    components are then independent.
48
49 Parameters bins, step and keep can also be entered as optional parameters.
50
51 <b>EXAMPLE</b>: I use the following settings
52
53 optim_s = Source_Optimizer(options="please be clever") (same as empty)
54 (...)
55 Monitor_Optimizer(xmin=-0.05, xmax=0.05, ymin=-0.05, ymax=0.05,
56 optim_comp = "optim_s")
57
58 A good optimization needs to record enough non optimized neutrons on
    Monitor
59 during step 2. Typical enhancement in computation speed is by a factor 20.
60 This component usually works well.
61
62 <b>NOTE:</b> You must be aware that in some cases (SetV and SetS),
63 the optimization might slightly affect the energy or spin distribution of the
64 source. The optimizer tries to do its best anyway.
65 Also, some 'spikes' may sometime appear in monitor signals in the course of
66 the optimization, coming from non-optimized neutrons with original weight.
67 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

Name	Unit	Description	Default
keep	0-100	Percentage of initial source distribution that is kept.	0.1
options	str	string of options. <b>Description<b>	See 0

## Links

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

## 3.9. The Monitor\_Optimizer McStas Component

To be used after the <b>Source\_Optimizer</b> component

### Identification

- **Site:**
- **Author:** <a href="mailto:farhi@ill.fr">Emmanuel Farhi</a>
- **Origin:** <a href="http://www.ill.fr">ILL (France)</a>
- **Date:** 17 Sept 1999

### Description

```

1 A component that optimizes the neutron flux passing through the
2 <b>Source_Optimizer</b> in order to have the maximum flux at the
3 Monitor_Optimizer position(s).
4 <b>Source_optimizer</b> should be placed just after the source.
5 Monitor_Optimizer should be placed at the position to optimize.
6 I prefer to put one just before the sample.
7
8 See <a href="Source_Optimizer.html">Source_Optimizer</a> for
9 usage example and additional informations.
```

### Input parameters

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

Name	Unit	Description	Default
<b>optim_comp</b>	str	name of the Source_Optimizer component in the instrument definition. Do not use quotes (no quotes)	

Name	Unit	Description	Default
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](Source_Optimizer.html)



### 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**) see section ??,
- ESS-project (**ESS\_moderator\_long** and **ESS\_moderator\_short**).

When no analytical model (e.g. a Maxwellian distribution) exists, one may have access to measurements, estimated flux distributions, event files, and - better - to MCNP/Tripoli4 neutron event records. The following components are then useful:

- **misc/Virtual\_input** can read a McStas event file (in text or binary format), often bringing an order-of-magnitude speed-up. See section 10.2.
- **contrib/Virtual\_tripoli4\_input** does the same, but from event files (text format) obtained from the *Tripoli4* [Tri] reactor simulation program. Such files are usually huge.
- **contrib/Virtual\_mcnp\_input** can read MCNP "PTRAC" event files (text format) obtained from the *MCNP* [Mcnb] reactor simulation program. Such files are usually huge.
- **misc/Vitess\_input** can read *Vitess* [Vit] neutron event binary files.
- **optics/Filter\_gen** reads a 1D distribution from a file, and may either modify or set the flux according to it. See section 4.4.

## 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

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

#### Description

```
1 An arm does not actually do anything, it is just there to set
2 up a new coordinate system.
3
4 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**.

## 4.2. The Slit McStas Component

Rectangular/circular slit

### Identification

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

### Description

```
1 A simple rectangular or circular slit. You may either
2 specify the radius (circular shape), or the rectangular bounds.
3 No transmission around the slit is allowed.
4
5 Example: Slit (xmin=-0.01, xmax=0.01, ymin=-0.01, ymax=0.01)
6 Slit (xwidth=0.02, yheight=0.02)
7 Slit (radius=0.01)
8
9 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
yheight	m	Height of slit. Overrides ymin,ymax if they are unset.	UNSET

### Links

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

### 4.3. The Beamstop McStas Component

Rectangular/circular beam stop.

#### Identification

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

#### Description

```
1 A simple rectangular or circular beam stop.  
2 Infinitely thin and infinitely absorbing.  
3 The beam stop is by default rectangular. You may either  
4 specify the radius (circular shape), or the rectangular bounds.  
5  
6 Example: Beamstop(xmin=-0.05, xmax=0.05, ymin=-0.05, ymax=0.05)  
7 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**.

File name	Description
Be.trm	Beryllium filter for cold neutron spectrometers (e.g. $k < 1.55 \text{ \AA}^{-1}$ )
HOPG.trm	Highly oriented pyrolytic graphite for $\lambda/2$ filtering (e.g. thermal beam at $k = 1.64, 2.662$ , and $4.1 \text{ \AA}^{-1}$ )
Sapphire.trm	Sapphire ( $\text{Al}_2\text{O}_3$ ) filter for fast neutrons ( $k > 6 \text{ \AA}^{-1}$ )

Table 4.4.: Some transmission data files to be used with e.g. the Filter\_gen component

#### 4.4. Filter\_gen: A general filter using a transmission table

This component is an ideal flat filter that changes the neutron flux according to a 1D input table (text file).

**Filter\_gen** may act as a source (*options*="set") or a filter (*options*="multiply", default mode). The table itself is a 2 column free format file which accept comment lines. The first table column represents wavevector, energy, or wavelength, as specified in the *options* parameter, whereas the second column is the transmission/weight modifier.

A usage example as a source would use *options*="wavelength, set", if the first column in the data is supposed to be  $\lambda$  (in  $\text{\AA}$ ). Another example using the component as a filter would be *options*="energy, multiply" if the first column is  $E$  (in meV).

The input parameters are the filter window size  $x_{min}, x_{max}, y_{min}, y_{max}$ , the behaviour specification string *options* and the file to use *file*. Additionally, rescaling can be made automatic with the *scaling* and relative *thickness* parameters. If for instance the transmission data file corresponds to a 5 cm thick filter, and one would like to simulate a 10 cm thick filter, then use *thickness* = 2.

Some example data files are given with McStas in the MCSTAS/data directory as \*.trm files for transmission.

The filter geometry is a flat plane. A geometry with finite thickness can be simulated by surrounding this component with two slits.

## 4.5. The Collimator\_linear McStas Component

A simple analytical Soller collimator (with triangular transmission).

### Identification

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

### Description

```
1 Soller collimator with rectangular opening and specified length. The
2 transmission function is an average and does not utilize knowledge of the
3 actual neutron trajectory. A zero divergence disables collimation (then the
4 component works as a double slit).
5
6 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 $\text{atan}(d/\text{length})$ , where d is the blade spacing)	40
transmission	1	Transmission of Soller ( $0 \leq t \leq 1$ )	1
divergenceV	minutes of arc	Divergence vertical angle	0

### Links

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

## 4.6. The Collimator\_radial McStas Component

A radial Soller collimator.

### Identification

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

### Description

```
1 Radial Soller collimator with rectangular opening and specified length.
2 The collimator is made of many rectangular channels stacked radially.
3 Each channel is a set of transmitting layers (nslit), separated by an
  absorbing
4 material (infinitely thin), the whole stuff is inside an absorbing housing.
5
6 When specifying the number of channels (nchan), each channel has a total
7 entrance width=radius*fabs(theta_max-theta_min)/nchan, but only the central
8 portion 'xwidth' accepts neutrons. When xwidth=0, it is set to the full
9 apperture so that all neutrons enter the channels (all walls are infinitely
  thin).
10
11 When using zero as the number of channels (nchan), the collimator is
  continuous,
12 without shadowing effect.
13
14 The component should be positioned at the radius center.
15 The component can be made oscillating (usual on diffractometers and TOF
16 machines) with the 'roc' parameter.
17 The neutron beam outside the collimator angular area is transmitted
  unaffected.
18
19 When used as a focusing collimator, the focusing parameter should be set to
  1.
20
21 An example of a instrument that uses this collimator can be found in the
  SALSA instrument,
22 in the example folder
23
24 Example:
25 Channelled radial collimator with shadow parts
26 Collimator_radial(xwidth=0.015, yheight=.3, length=.35, divergence=40,
  transmission=1, theta_min=5, theta_max=165, nchan=128, radius=0.9)
27 A continuous radial collimator
28 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

## Links

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



## 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. The Guide McStas Component

Neutron guide.

#### Identification

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

#### Description

```
1 Models a rectangular guide tube centered on the Z axis. The entrance lies
2 in the X-Y plane.
3 For details on the geometry calculation see the description in the McStas
4 reference manual.
5 The reflectivity profile may either use an analytical mode (see Component
6 Manual) or a 2-columns reflectivity free text file with format
7 [q(Angs-1) R(0-1)].
8
9 Example: Guide(w1=0.1, h1=0.1, w2=0.1, h2=0.1, l=2.0, R0=0.99, Qc=0.021,
10           alpha=6.07, m=2, W=0.003
11 %VALIDATION
12 May 2005: extensive internal test, no bugs found
13 Validated by: K. Lieutenant
14
15 %BUGS
16 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(Angs-1) R(0-1)>	0
<b>w1</b>	m	Width at the guide entry	
<b>h1</b>	m	Height at the guide entry	
w2	m	Width at the guide exit	0
h2	m	Height at the guide exit	0
<b>l</b>	m	length of guide	
R0	1	Low-angle reflectivity	0.99
Qc	AA-1	Critical scattering vector	0.0219
alpha	AA	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	AA-1	Width of supermirror cut-off	0.003

## Links

- Component source code found in file **Guide.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

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

#### Description

```
1 Models a disc chopper with nslit identical slits , which are symmetrically
   distributed
2 on the disc . At time t=0, the centre of the first slit opening will be
   situated at the
3 vertical axis when phase=0, assuming the chopper centre of rotation is
   placed <b>BELOW</b> the beam axis .
4 If you want to place the chopper <b>ABOVE</b> the beam axis , please use a
   180 degree rotation around Z
5 (otherwise unexpected beam splitting can occur in combination with the
   isfirst=1 setting , see
6 <a href="https://github.com/mccode-dev/McCode/issues/650">related bug on
   GitHub</a>)
7
8 For more complicated gemometries , see component manual example of
   DiskChopper GROUPing.
9
10 If the chopper is the 1st chopper of a continuous source instrument , you
    should use the "isfirst" parameter .
```

```

11 This parameter SETS the neutron time to match the passage of the chooper
    slit(s), taking into account the
12 chopper timing and phasing (thus conserving your simulated statistics).
13
14 The isfirst parameter is ONLY relevant for use in continuous source
    settings.
15
16 Example: DiskChopper(radius=0.2, theta_0=10, nu=41.7, nslit=3, delay=0,
    isfirst=1) First chopper
17 DiskChopper(radius=0.2, theta_0=10, nu=41.7, nslit=3, delay=0, isfirst=0)
18
19 NOTA BENE wrt. GROUPing and isfirst:
20 When setting up a GROUP of DiskChoppers for a steady-state / reactor source
    , you will need
21 to set up
22 1) An initial chopper with isfirst=1, NOT part of the GROUP — and using a "
    big" chopper opening
23 that spans the full angular extent of the openings of the subsequent GROUP
24 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
<b>yheight</b>	m	Slit height (if = 0, equal to radius). Auto centering of beam at half height.	
<b>nu</b>	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

Name	Unit	Description	Default
verbose	1	Set to 1 to display Disk chopper configuration	0

## Links

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

## 6.2. The FermiChopper McStas Component

Fermi Chopper with rotating frame.

### Identification

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

### Description

```
1 Models a fermi chopper with optional supermirror coated blades
2 supermirror facilities may be disabled by setting m = 0, R0=0
3 Slit packages are straight. Chopper slits are separated by an infinitely
4 thin absorbing material. The effective transmission (resulting from
   fraction
5 of the transparent material and its transmission) may be specified.
6 The chopper slit package width may be specified through the total width '
   xwidth'
7 of the full package or the width 'w' of each single slit. The other
   parameter
8 is calculated by: xwidth = nslit*w. The slit package may be made curved and
   use
9 super-mirror coating.
10
11 Example:
12 FermiChopper(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)
13
14 %VALIDATION
15 Apr 2005: extensive external test, most problems solved (cf. 'Bugs')
16 Validated by: K. Lieutenant, E. Farhi
17
18 limitations:
19 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	AA-1	critical scattering vector	0.02176
alpha	AA	slope of reflectivity	2.33
m	1	m-value of material. Zero means completely absorbing.	0.0
W	AA-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, 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 transmission is centered on 'delay'	0

## Links

- Component source code found in file **FermiChopper.comp**.
- [Vitess\\_ChopperFermi](Vitess_ChopperFermi.html) component by
- G. Zsigmond, imported from Vitess by K. Lieutenant.

## 6.3. The Vitess\_ChopperFermi McStas Component

Fermi chopper with absorbing walls using the VITESS module 'chopper\_fermi'

### Identification

- **Site:**
- **Author:** Geza Zsigmond

- **Origin:** VITESS module 'chopper\_fermi'
- **Date:** Sep 2004

## Description

```

1 This component simulates a Fermi chopper with absorbing walls. The rotation
  axis is
2 vertical (y-axis), i.e. the path length through the channels is given by
  the length
3 'depth' along the z-axis.
4 The shape of the channels can be straight, curved with circular, or curved
  with ideal
5 (i.e. close to a parabolic) shape. This is determined by the parameter '
  GeomOption'.
6
7 Geometry for straight and circular channels:
8 The geometry of the chopper consists of a rectangular shaped object with a
  channel
9 system. In transmission position, there are 'Nchannels' slits along the x-
  axis,
10 separated by absorbing walls of thickness 'wallwidth' giving a total width
  'width'.
11 The rectangular channel system is surrounded by a so-called shadowing
  cylinder
12 (cf component manual).
13
14 Geometry for parabolic channels:
15 In this case, the Fermi chopper is supposed to be a full cylinder, i.e. the
  central
16 channels are longer than those on the edges (cf. figure in the component
  manual).
17 The other features are the same as for the other options.
18
19 Apart from the frequency of rotation, the phase of the chopper at t=0 has
  to be given;
20 phase = 0 means transmission orientation.
21
22 The option 'zerotime' may be used to reset the time at the chopper position
  . The
23 consequence is that only 1 pulse is generated instead of several.
24
25 NOTE: This component must NOT be located at the same position as the
  previous one.
26 This also stands for monitors and Arms. A non zero distance must be defined
  .
27
28 Examples:
29 straight Fermi chopper, 18000 rpm, 20 channels a 0.9 mm separated by 0.1 mm
  walls,
30 16 mm channel length, minimal shadowing cylinder, phased to be open at 1 ms
  ,

```



```

31 generation of only 1 pulse, normal precision (for short wavelength neutrons
   )
32 Vitess_ChopperFermi(GeomOption=0, zerotime=1, Nchannels=20, Ngates=4,
33 freq=300.0, height=0.06, width=0.0201,
34 depth=0.016, r_curv=0.0, diameter=0.025691, Phase=-108.0,
35 wallwidth=0.0001, sGeomFileName="FC_geom_str.dat")
36
37 Fermi chopper with circular channels, 12000 rpm, optimized for 6 Ang,
   several pulses,
38 highest accuracy (because of long wavelength neutrons used), rest as above
39 Vitess_ChopperFermi(GeomOption=2, zerotime=0, Nchannels=20, Ngates=8,
40 freq=200.0, height=0.06, width=0.0201,
41 depth=0.016, r_curv=0.2623, diameter=0.025691, Phase=-72.0,
42 wallwidth=0.0001, sGeomFileName="FC_geom_circ.dat")
43
44 %VALIDATION
45 Apr 2005: extensive external test, most problems solved (cf. 'Bugs' and
   source header)
46 Validated by: K. Lieutenant
47
48 limitations: slow (10 times slower than FermiChopper), especially for a
   high number of channels
49
50 %BUGS
51 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](http://www.hmi.de/projects/ess/vitess/DOC/chopper_fermi_str.html)
- [curved VITESS Fermi chopper](http://www.hmi.de/projects/ess/vitess/DOC/chopper_fermi_cur.html)

## 6.4. The V\_selector McStas Component

Velocity selector.

### Identification

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

### Description

```
1 Velocity selector consisting of rotating Soller-like blades
2 defining a helically twisted passage.
3 Geometry defined by two identical, centered apertures at 12 o'clock
4 position, Origo is at the centre of the selector (input is at -zdepth/2).
5 Transmission is analytical assuming a continuous source.
6
7 Example: V_selector(xwidth=0.03, yheight=0.05, zdepth=0.30, radius=0.12,
8             alpha=48.298, length=0.25, d=0.0004, nu=20000, nslit=72)
9 These are values for the D11@ILL Dornier 'Dolores' Velocity Selector (NVS
10 023)
11
12 %VALIDATION
13 Jun 2005: extensive external test, no problems found
14 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

Name	Unit	Description	Default
nu	Hz	Cylinder rotation speed, counter-clockwise, which is ideally $3956 \cdot \alpha \cdot \text{DEG2RAD} / 2 / \text{PI} / \text{length}$	300
nslit	1	Number of Soller blades	72

## Links

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

## 6.5. The Selector McStas Component

velocity selector (helical lamella type) such as `<b>V_selector</b>` component

### Identification

- **Site:**
- **Author:** Peter Link, <mailto:Andreas.Ostermann@frm2.tum.de> Andreas Ostermann
- **Origin:** Uni. Gottingen (Germany)
- **Date:** MARCH 1999

### Description

```

1 Velocity selector consisting of rotating Soller-like blades
2 defining a helically twisted passage.
3 Geometry is defined by two identical apertures at 12 o'clock position,
4 The origin is at the ENTRANCE of the selector.
5
6 Example: Selector(xmin=-0.015, xmax=0.015, ymin=-0.025, ymax=0.025,
7           length=0.25,
8           nslit=72,d=0.0004, radius=0.12, alpha=48.298, nu=500)
9 These are values for the D11@ILL Dornier 'Dolores' Velocity Selector (NVS
10 023)
11
12 %VALIDATION
13 Jun 2005: extensive external test, one minor problem
14 Jan 2006: problem solved (for McStas-1.9.1)
15 Validated by: K. Lieutenant
16
17 %BUGS
18 for transmission calculation, each neutron is supposed to be in the guide
19 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 $3956 \cdot \alpha$ - $\text{pha} \cdot \text{DEG2RAD} / 2 / \text{PI} / \text{lambda} / \text{length}$	500

## Links

- Component source code found in file **Selector.comp**.
- See also Additional notes <http://mcstas.risoe.dk/pipermail/neutron-mc/1999q1/000134.html> >M. 1999</a> and <http://mcstas.risoe.dk/pipermail/neutron-mc/1999q2/000136.html> >Jan 2000</a>.

## 7. 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 8.

### 7.1. The Monochromator\_flat McStas Component

Flat Monochromator crystal with anisotropic mosaic.

#### Identification

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

#### Description

```
1 Flat, infinitely thin mosaic crystal, useful as a monochromator or analyzer
2
3 For an unrotated monochromator component, the crystal surface lies in the Y
  -Z
4 plane (ie. parallel to the beam).
5 The mosaic is anisotropic gaussian, with different FWHMs in the Y and Z
6 directions. The scattering vector is perpendicular to the surface.
7
8 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)
9
10 Monochromator lattice parameter
11 PG      002 DM=3.355 AA (Highly Oriented Pyrolythic Graphite)
12 PG      004 DM=1.677 AA
13 Heusler 111 DM=3.362 AA (Cu2MnAl)
14 CoFe    DM=1.771 AA (Co0.92Fe0.08)
15 Ge      111 DM=3.266 AA
```

```

15 Ge      311 DM=1.714 AA
16 Ge      511 DM=1.089 AA
17 Ge      533 DM=0.863 AA
18 Si      111 DM=3.135 AA
19 Cu      111 DM=2.087 AA
20 Cu      002 DM=1.807 AA
21 Cu      220 DM=1.278 AA
22 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 min-utes	Horizontal mosaic (in z direction) (FWHM)	30.0
mosaicv	arc min-utes	Vertical mosaic (in y direction) (FWHM)	30.0
r0	1	Maximum reflectivity	0.7
Q	1/angstrom	Magnitude of scattering vector	1.8734
DM	AA	monochromator d-spacing, instead of Q = 2*pi/DM	0

## Links

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

## 7.2. The Monochromator\_curved McStas Component

Double bent multiple crystal slabs with anisotropic gaussian mosaic.

### Identification

- **Site:**
- **Author:** Emmanuel Farhi, Kim, Lefmann, Peter Link

- **Origin:** <http://www.ill.fr> >ILL</a>

- **Date:** Aug. 24th 2001

## Description

```
1 Double bent infinitely thin mosaic crystal, useful as a monochromator or
2 analyzer. which uses a small-mosaicity approximation and taking into
   account
3 higher order scattering. The mosaic is anisotropic gaussian, with
   different
4 FWHMs in the Y and Z directions. The scattering vector is perpendicular to
   the
5 surface. For an unrotated monochromator component, the crystal plane lies
   in
6 the y-z plane (ie. parallel to the beam). The component works in reflection
   , but
7 also transmits the non-diffracted beam. Reflectivity and transmission files
   may
8 be used. The slabs are positioned in the vertical plane (not on a
9 cylinder/sphere), and are rotated according to the curvature radius.
10 When curvatures are set to 0, the monochromator is flat.
11 The curvatures approximation for parallel beam focusing to distance L, with
12 monochromator rotation angle A1 are:
13 RV = 2*L*sin(DEG2RAD*A1);
14 RH = 2*L/sin(DEG2RAD*A1);
15
16 When you rotate the component by A1 = asin(Q/2/Ki)*RAD2DEG, do not forget
   to
17 rotate the following components by A2=2*A1 (for 1st order) !
18
19 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)
20
21 Monochromator lattice parameter
22 PG      002 DM=3.355 AA (Highly Oriented Pyrolythic Graphite)
23 PG      004 DM=1.677 AA
24 Heusler 111 DM=3.362 AA (Cu2MnAl)
25 CoFe    DM=1.771 AA (Co0.92Fe0.08)
26 Ge      111 DM=3.266 AA
27 Ge      311 DM=1.714 AA
28 Ge      511 DM=1.089 AA
29 Ge      533 DM=0.863 AA
30 Si      111 DM=3.135 AA
31 Cu      111 DM=2.087 AA
32 Cu      002 DM=1.807 AA
33 Cu      220 DM=1.278 AA
34 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"
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 min-utes	Horizontal mosaic FWHM	30.0
mosaicv	arc min-utes	Vertical mosaic FWHM	30.0
r0	1	Maximum reflectivity. O unactivates component	0.7
t0	1	transmission efficiency	1.0
Q	AA-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	AA	monochromator d-spacing instead of $Q=2\pi/DM$	0
mosaic	arc min-utes	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 preferred. Use 0 for all	0

## Links

- Component source code found in file `Monochromator_curved.comp`.
- <http://mcstas.risoe.dk/pipermail/neutron-mc/1999q1/000133.html> Additional note from Peter Link.
- Obsolete Mosaic\_anisotropic by Kristian Nielsen
- Contributed Monochromator\_2foc by Peter Link

### 7.3. 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 ??.

### 7.4. 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 SPPEED 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.

## 8. 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 simple component **V-sample** performs both incoherent scattering and absorption.

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 environments.

Next type is Bragg scattering from single crystals. The simplest single crystals are in fact the monochromator components like **Monochromator\_flat**, presented in section ???. 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.

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.

### 8.0.1. Neutron scattering notation

In sample components, we use the notation common for neutron scattering, where the wave vector transfer is denoted the *scattering vector*

$$\mathbf{q} \equiv \mathbf{k}_i - \mathbf{k}_f. \quad (8.1)$$

Sample Process	Coherent		Incoherent		Absorption	Multi. Scatt.
	Elastic	Inelastic	Elastic	Inelastic		
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 8.1.: Processes implemented in sample components. Absorption: 1=single only, 2=with secondary

In analygo, the *energy transfer* is given by

$$\hbar\omega \equiv E_i - E_f = \frac{\hbar^2}{2m_n} (k_i^2 - k_f^2). \quad (8.2)$$

### 8.0.2. 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  $\sigma_c^a$  and a scattering cross section per unit cell of  $\sigma_c^s$ . 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 8.1. 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

$$f_{\text{att}}(l) = \exp(-l(\mu^s + \mu^a)), \quad (8.3)$$

where the normalization  $f_{\text{att}}(0) = 1$ .

The probability for a given neutron ray to be scattered from within the interval  $[l_1; l_1 + dl]$  will be

$$P(l_1)dl = \mu^s f_{\text{att}}(l_1)dl, \quad (8.4)$$

while the probability for a neutron to be scattered from within this interval into the solid angle  $\Omega$  and not being scattered further or absorbed on the way out of the sample is

$$P(l_1, \Omega)dld\Omega = \mu^s f_{\text{att}}(l_1)f_{\text{att}}(l_2)\gamma(\Omega)d\Omega dl, \quad (8.5)$$

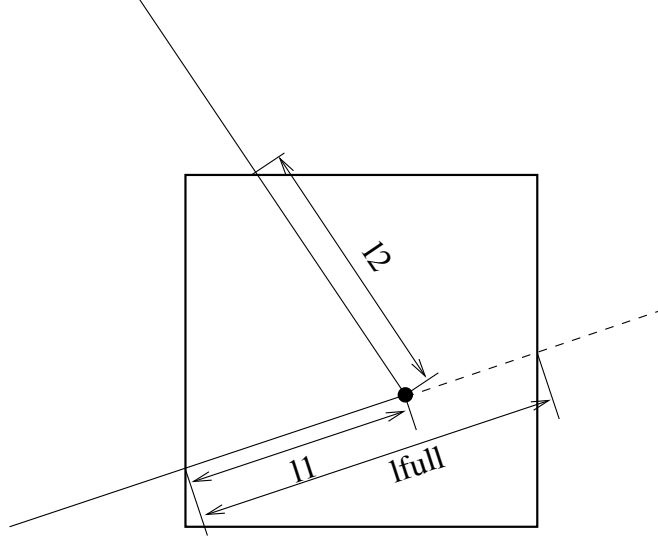


Figure 8.1.: The geometry of a scattering event within a powder sample.

where  $\gamma(\Omega)$  is the directional distribution of the scattered neutrons, and  $l_2$  is determined by Monte Carlo choices of  $l_1$ ,  $\Omega$ , and from the sample geometry, see e.g. figure 8.1.

In our Monte-Carlo simulations, we may choose the scattering parameters by making a Monte-Carlo choice of  $l_1$  and  $\Omega$  from a distribution different from  $P(l_1, \Omega)$ . By doing this, we must adjust  $\pi_i$  according to the probability transformation rule (??). 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

$$f(l_1, \Omega) = g(\Omega)/l_{\text{full}}, \quad (8.6)$$

$l_{\text{full}}$  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 (??), the neutron weight factor is now adjusted by the amount

$$\pi_i(l_1, \Omega) = \mu^s l_{\text{full}} \exp [-(l_1 + l_2)(\mu^a + \mu^s)] \frac{\gamma(\Omega)}{g(\Omega)}. \quad (8.7)$$

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 (8.7), 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.

### 8.0.3. Future development of sample components

There is still room for much more development of functionality in McStas samples.

A more general SANS sample is under development. In addition, a reflectometry sample will soon be developed. In the mean time, you may use the **SiC** contributed component.

In general, all samples are assumed to be homogeneous. There would also be potential in developing an inhomogeneous sample, e.g. with spatially varying lattice constant, relevant for stress/strain scanners. Inhomogeneously absorbing sample for tomography could also be possible. Further, no polarization effects are yet taken into account in any of the samples.

## 8.1. The Incoherent McStas Component

Incoherent sample (such as Vanadium) sample, with quasielastic component OR or global energy transfer.

### Identification

- **Site:**
- **Author:** Kim Lefmann and Kristian Nielsen
- **Origin:** Risoe
- **Date:** 15.4.98

### Description

```
1 A Double—cylinder shaped incoherent scatterer (like Vanadium)
2 with both elastic and quasielastic (Lorentzian) components.
3 No multiple scattering (but approximation available). Absorption included.
4 <b>Sample focusing:</b>
5 The area to scatter to is a disk of radius 'focus_r' situated at the target
6
7 This target area may also be rectangular if specified focus_xw and focus_yh
8 or focus_aw and focus_ah, respectively in meters and degrees.
9 The target itself is either situated according to given coordinates (x,y,z)
10
11 or defined with the relative target_index of the component to focus
12 to (next is +1).
13 This target position will be set to its AT position. When targeting to
14 centered components, such as spheres or cylinders, define an Arm component
15 where to focus to.
16 <b>Sample shape:</b>
17 Sample shape may be a cylinder, a sphere, a box or any other shape
18 box/plate:          xwidth x yheight x zdepth (thickness=0)
19 hollow box/plate: xwidth x yheight x zdepth and thickness>0
20 cylinder:           radius x yheight (thickness=0)
```

```

19 hollow cylinder: radius x yheight and thickness>0
20 sphere:         radius (yheight=0 thickness=0)
21 hollow sphere:  radius and thickness>0 (yheight=0)
22 any shape:      geometry=OFF file
23
24 The complex geometry option handles any closed non-convex polyhedra.
25 It computes the intersection points of the neutron ray with the object
26 transparently, so that it can be used like a regular sample object.
27 It supports the PLY, OFF and NOFF file format but not COFF (colored faces).
28 Such files may be generated from XYZ data using:
29 qhull < coordinates.xyz Qx Qv Tv o > geomview.off
30 or
31 powercrust coordinates.xyz
32 and viewed with geomview or java -jar jroff.jar (see below).
33 The default size of the object depends of the OFF file data, but its
34 bounding box may be resized using xwidth,yheight and zdepth.
35
36 Example: Incoherent(radius=0.05,focus_r=0.035, pack=1, target_index=1)
37 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 Object File Format (OFF) or PLY file for complex geometry. The OFF/PLY file may be generated from XYZ coordinates using qhull/powercrust	0
radius	m	Outer radius of sample in (x,z) plane	0
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			0
target_y	m	position of target to focus at	0
target_z			0
focus_r	m	Radius of disk containing target. Use 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

Name	Unit	Description	Default
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	AA <sup>3</sup>	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
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**.
- <http://www.ncnr.nist.gov/resources/n-lengths/>>Cross sections for single elements</a>
- <http://www.ncnr.nist.gov/resources/sldcalc.html>>Cross sections for compounds</a>
- <http://www.webelements.com/>>Web Elements</a>
- <http://neutron.risoe.dk/mcstas/components/tests/Incoherent/>>Test results</A> (not up-to-date).



- The test/example instrument [vanadium\\_example.instr]( ../examples/vanadium_example.instr)
- The test/example instrument [QENS\\_test.instr]( ../examples/QENS_test.instr).
- [Geomview and Object File Format \(OFF\)](http://www.geomview.org)
- Java version of Geomview (display only) [jroff.jar](http://www.holmes3d.net/graphics/roffview/)
- [qhull](http://qhull.org)
- [powercrust](http://www.cs.ucdavis.edu/~amenta/powercrust.html)

## 8.2. The Tunneling\_sample McStas Component

A Double-cylinder shaped all-incoherent scatterer with elastic, quasielastic (Lorentzian), and tunneling (sharp) components.

### Identification

- **Site:**
- **Author:** Kim Lefmann
- **Origin:** Risoe
- **Date:** 10.05.07

### Description

```
1 A Double-cylinder shaped all-incoherent scatterer
2 with both elastic, quasielastic (Lorentzian), and tunneling (sharp)
3 components. No multiple scattering. Absorbtion included.
4 The shape of the sample may be a box with dimensions xwidth, yheight,
   zdepth.
5 The area to scatter to is a disk of radius 'focus_r' situated at the target
   .
6 This target area may also be rectangular if specified focus_xw and focus_yh
7 or focus_aw and focus_ah, respectively in meters and degrees.
8 The target itself is either situated according to given coordinates (x,y,z)
   ,
9 or defined with the relative target_index of the component to focus
10 to (next is +1).
11 This target position will be set to its AT position. When targeting to
12 centered components, such as spheres or cylinders, define an Arm component
13 where to focus to.
14
15 The outgoing polarization is calculated as for nuclear spin incoherence:
16  $P' = 1/3 * P - 2/3 P = -1/3 P$ 
17 As above multiple scattering is ignored .
18
19 Example: Tunneling_sample(thickness=0.001, radius=0.01, yheight=0.02, focus_r
   =0.035,
20 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
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	Absorption cross section pr. unit cell	5.08
sigma_inc	barns	Total incoherent scattering cross section pr. unit cell	4.935
Vc	AA <sup>3</sup>	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`.
- [http://neutron.risoe.dk/mcstas/components/tests/v\\_sample/](http://neutron.risoe.dk/mcstas/components/tests/v_sample/) Test results (not up-to-date).

## 8.3. The PowderN McStas Component

General powder sample (N lines, single scattering, incoherent scattering)

### Identification

- **Site:**
- **Author:** P. Willendrup, L. Chapon, K. Lefmann, A.B.Abrahamsen, N.B.Christensen, E.M.Lauridsen.
- **Origin:** McStas release
- **Date:** 4.2.98

### Description

```
1 General powder sample with
2 many scattering vectors
3 possibility for intrinsic line broadening
4 incoherent elastic background ratio is specified by user
5 No multiple scattering. No secondary extinction.
6
7 Based on Powder1/Powder2/Single_crystal.
8 Geometry is a powder filled cylinder, sphere, box or any shape from an OFF
  file.
9 Incoherent scattering is only provided here to account for a background.
10 The efficient is highly improved when restricting the vertical scattering
11 range on the Debye-Scherrer cone (with 'd_phi' and 'focus_flip').
12 The unit cell volume Vc may also be computed when giving the density,
13 the atomic/molecular weight and the number of atoms per unit cell.
14 A simple strain handling is available by mean of either a global Strain
  parameter,
15 or a column with a strain value per Bragg reflection. The strain values are
16 specified in ppm (1e-6).
17 The Single_crystal component can also handle a powder mode, as well as an
18 approximated texture.
19
20 <b>Sample shape:</b>
21 Sample shape may be a cylinder, a sphere, a box or any other shape.
22 box/plate:      xwidth x yheight x zdepth (thickness=0)
23 hollow box/plate: xwidth x yheight x zdepth and thickness>0
24 cylinder:      radius x yheight (thickness=0)
25 hollow cylinder: radius x yheight and thickness>0
26 sphere:        radius (yheight=0 thickness=0)
27 hollow sphere: radius and thickness>0 (yheight=0)
28 any shape:      geometry=OFF_file
29
30 The complex geometry option handles any closed non-convex polyhedra.
31 It computes the intersection points of the neutron ray with the object
32 transparently, so that it can be used like a regular sample object.
33 It supports the PLY, OFF and NOFF file format but not COFF (colored faces).
```

```

34 Such files may be generated from XYZ data using:
35 qhull < coordinates.xyz Qx Qv Tv o > geomview.off
36 or
37 powercrust coordinates.xyz
38 and viewed with geomview or java -jar jroff.jar (see below).
39 The default size of the object depends of the OFF file data, but its
40 bounding box may be resized using xwidth,yheight and zdepth.
41
42 If you use this component and produce valuable scientific results, please
43 cite authors with references bellow (in <a href="#links">Links</a>).
44
45 Example: PowderN(reflections = "c60.lau", d_phi = 15 , radius = 0.01,
46 yheight = 0.05, Vc = 1076.89, sigma_abs = 0, delta_d_d=0, DW=1))
47
48 <b>Powder definition file format</b>
49 Powder structure is specified with an ascii data file 'reflections'.
50 The powder data are free-text column based files.
51 The reflection list should be ordered by decreasing d-spacing values.
52 ... d ... F2
53 Lines begining by '#' are read as comments (ignored) but they may contain
54 the following keywords (in the header):
55 #Vc <value of unit cell volume Vc [Angs^3]>
56 #sigma_abs <value of Absorption cross section [barns]>
57 #sigma_inc <value of Incoherent cross section [barns]>
58 #Debye_Waller <value of Debye-Waller factor DW>
59 #delta_d_d/d <value of delta_d_d/d width for all lines>
60 These values are not read if entered as component parameters (Vc=...)
61
62 The signification of the columns in the numerical block may be
63 set using the 'format' parameter, by defining signification of the
64 columns as a vector of indexes in the order
65 format={j,d,F2,DW,delta_d_d/d,1/2d,q,F,Strain}
66
67 Signification of the symbols is given below. Indices start at 1.
68 Indices with zero means that the column are not present, so that:
69 Crystallographica={ 4,5,7,0,0,0,0,0,0 }
70 Fullprof = { 4,0,8,0,0,5,0,0,0 }
71 Lazy = {17,6,0,0,0,0,0,13,0}
72
73 At last, the format may be overridden by direct definition of the
74 column indexes in the file itself by using the following keywords
75 in the header (e.g. '#column_j 4'):
76 #column_j <index of the multiplicity 'j' column>
77 #column_d <index of the d-spacing 'd' column [Angs]>
78 #column_F2 <index of the squared str. factor '|F|^2' column [b]>
79 #column_F <index of the structure factor norm '|F|' column>
80 #column_DW <index of the Debye-Waller factor 'DW' column>
81 #column_Dd <index of the relative line width delta_d_d/d broadening 'Dd'
column>
82 #column_inv2d <index of the 1/2d=sin(theta)/lambda 'inv2d' column>
83 #column_q <index of the scattering wavevector 'q' column [Angs-1]>
84 #column_strain <index of the strain line shift Delta/d [ppm]>
85

```

```

86 Last, CIF, FullProf and ShelX files can be read, and converted to F2(hkl)
    lists
87 if 'cif2hkl' is installed. The CIF2HKL env variable can be used to point to
    a
88 proper executable, else the McCode, then the system installed versions are
    used.
89
90 <b>Concentricity</b>
91
92 PowderN assumes 'concentric' shape, i.e. can contain other components
    inside its
93 optional inner hollow. Example, Sample in Al cryostat:
94
95
96 COMPONENT Cryo = PowderN(reflections="Al.laz", radius = 0.01, thickness =
    0.001,
97 concentric = 1, p_interact=0.1)
98 AT (0,0,0) RELATIVE Somewhere
99
100 COMPONENT Sample = some_other_component(with geometry FULLY enclosed in the
    hollow)
101 AT (0,0,0) RELATIVE Somewhere
102
103 COMPONENT Cryo2 = COPY(Cryo)(concentric = 0)
104 AT (0,0,0) RELATIVE Somewhere
105
106
107 (The second instance of the cryostat component can also be written out
    completely
108 using PowderN(...). In both cases, this second instance needs concentric =
    0.)
109 The concentric arrangment can not be used with OFF geometry specification.
110
111 This sample component can advantageously benefit from the SPLIT feature, e.
    g.
112 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 Object File Format (OFF) or PLY file for complex geometry. The OFF/PLY file may be generated from XYZ coordinates using qhull/powercrust.	"NULL"

Name	Unit	Description	Default
format	no quotes	Name of the format, or list of column indexes (see Description).	{0, 0, 0, 0, 0, 0, 0, 0}
radius	m	Outer radius of sample in (x,z) plane	0
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	AA <sup>3</sup>	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 <sup>3</sup>	Density of material. $\rho = \text{density} / \text{weight} / 1e24 * N\_A$ .	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 <sup>2</sup> (barns=1 for laz, barns=0 for lau type files).	1
Strain	ppm	Global relative $\Delta d/d$ 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
target_index	1	Relative index of component to focus incoherent scattering at, e.g. next is +1	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 <http://icsd.ill.fr> ICSD Inorganic Crystal Structure Database
- <http://www.ncnr.nist.gov/resources/n-lengths/> Cross sections for single elements
- <http://www.ncnr.nist.gov/resources/sldcalc.html> Cross sections for compounds
- <http://www.webelements.com/> Web Elements
- <http://www.ill.eu/sites/fullprof/index.html> Fullprof powder refinement
- <http://www.crystallographica.com/> Crystallographica software (free license)



- [Geomview and Object File Format \(OFF\)](http://www.geomview.org)
- Java version of Geomview (display only) [jroff.jar](http://www.holmes3d.net/graphics/roffview/)
- [qhull](http://qhull.org)
- [powercrust](http://www.cs.ucdavis.edu/~amenta/powercrust.html)

## 8.4. The Single\_crystal McStas Component

Mosaic single crystal with multiple scattering vectors, optimised for speed with large crystals and many reflections.

### Identification

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

### Description

```
1 Single crystal with mosaic. Delta-D/D option for finite-size effects.
2 Rectangular geometry. Multiple scattering and secondary extinction included
3
4 The mosaic may EITHER be specified isotropic by setting the mosaic input
5 parameter, OR anisotropic by setting the mosaic_a, mosaic_b, and mosaic_c
6 parameters.
7 The crystal lattice can be bent locally, keeping the external geometry
8 unchanged.
9 Curvature is spherical along vertical and horizontal axes.
10
11 <b>Speed/stat optimisation using SPLIT</b>
12 In order to dramatically improve the simulation efficiency, we recommend to
13 use a SPLIT keyword on this component (or prior to it), as well as to
14 disable
15 the multiple scattering handling by setting order=1. This is especially
16 powerful
17 for large reflection lists such as with macromolecular proteins. When an
18 incoming
19 particle is identical to the preceeding, reciprocal space initialisation is
20 skipped, and a Monte Carlo choice is done on available reflections from the
21 last
22 reciprocal space calculation! To assist the user in choosing a "relevant"
23 value
24 of the SPLIT, a rolling average of the number of available reflections is
25 calculated and presented in the component output.
26
27 <b>Mosacitiy modes:</b>
28 The component features three independent ways of parametrising mosaicity:
29 a) The original algorithm where mosaicity is implemented by extending each
30 reflection by a Gaussian "cigar" in reciprocal space, characterised by
31 the parameters mosaic and delta_d_d.
32 (Also known as "isotropic mosaicity".)
33 b) A similar mode where mosaicities can be non-isotropic and given as the
34 parameters mosaic_a, mosaic_b and mosaic_c, around the unit cell axes.
35 (Also known as "anisotropic mosaicity".)
```

29 c) Given two "macroscopically"/experimentally measured width/mosaicitities  
 30 of two independent reflections, parametrised by the list  
 31 mosaic\_AB = {mos\_a, mos\_b, a\_h, a\_k, a\_l, b\_h, b\_k, b\_l}, a set of  
 32 microscopic mosaicities as in b) are estimated (internally) **and** applied.  
 33 (Also known as "phenomenological mosaicity".)  
 34  
 35 <b>Powder— **and** PG-mode</b>  
 36 When these two modes are used (powder=1 **or** PG=1), a randomised  
 transformation  
 37 of the particle direction is made before **and** after scattering, thereby  
 letting  
 38 the single crystal behave as a crystallite of either a powder (crystallite  
 39 orientation fully randomised) **or** pyrolytic graphite (crystallite randomised  
 around  
 40 the c-axis).  
 41  
 42 <b>Curved crystal mode</b>  
 43 The component features a method to curve the lattice planes slightly with  
 respect  
 44 to the outer geometry of the crystal. The method is implemented as a  
 transformation  
 45 on the particle direction vector, **and** should be used only in cases where  
 46 a) The reflection lattice vector is  $\sim$  orthogonal to the crystal surface  
 47 b) The modelled curvature is "small" with respect to the crystal surface  
 48  
 49 <b>Sample shape:</b>  
 50 Sample shape may be a cylinder, a sphere, a box **or** any other shape  
 51 box/plate:        xwidth x yheight x zdepth  
 52 cylinder:        radius x yheight  
 53 sphere:        radius (yheight=0)  
 54 any shape:        geometry=OFF file  
 55  
 56 The complex geometry option handles any closed non-convex polyhedra.  
 57 It computes the intersection points of the neutron ray with the object  
 58 transparently, so that it can be used like a regular sample object.  
 59 It supports the PLY, OFF **and** NOFF file format but **not** COFF (colored faces).  
 60 Such files may be generated from XYZ data **using**:  
 61 qhull < coordinates.xyz Qx Qv Tv o > geomview.off  
 62 **or**  
 63 powercrust coordinates.xyz  
 64 **and** viewed with geomview **or** java -jar jroff.jar (see below).  
 65 The **default** size of the object depends on the OFF file data, but its  
 66 bounding box may be resized **using** xwidth,yheight **and** zdepth.  
 67  
 68 <b>Crystal definition file format</b>  
 69 Crystal structure is specified with an ascii data file. Each line contains  
 70 4 **or** more numbers, separated by white spaces:  
 71  
 72 h k l ... F2  
 73  
 74 The first three numbers are the (h,k,l) indices of the reciprocal lattice  
 75 point, **and** the 7-th number is the value of the structure factor  $|F|^{*2}$ , in  
 76 barns. The rest of the numbers are **not** used; the file is in the format  
 77 output by the Crystallographica program.

```

78 The reflection list should be ordered by decreasing d-spacing values.
79 Lines begining by '#' are read as comments (ignored). Most sample
    parameters
80 may be defined from the data file header, following the same mechanism as
81 PowderN.
82
83 Current data file header keywords include, for data format specification:
84 #column_h <index of the Bragg Qh column>
85 #column_k <index of the Bragg Qk column>
86 #column_l <index of the Bragg Ql column>
87 #column_F2 <index of the squared str. factor '|F|^2' column [b]>
88 #column_F <index of the structure factor norm '|F|' column>
89 and for material specification:
90 #sigma_abs <value of absorption cross section [barns]>
91 #sigma_inc <value of incoherent cross section [barns]>
92 #Delta_d/d <value of Delta_d/d width for all lines>
93 #lattice_a <value of the a lattice parameter [Angs]>
94 #lattice_b <value of the b lattice parameter [Angs]>
95 #lattice_c <value of the c lattice parameter [Angs]>
96 #lattice_aa <value of the alpha lattice angle [deg]>
97 #lattice_bb <value of the beta lattice angle [deg]>
98 #lattice_cc <value of the gamma lattice angle [deg]>
99
100 Last, CIF, FullProf and ShelX files can be read, and converted to F2(hkl)
    lists
101 if 'cif2hkl' is installed. The CIF2HKL env variable can be used to point to
    a
102 proper executable, else the McCode, then the system installed versions are
    used.
103
104 See the Component Manual for more details.
105
106 Example: Single_crystal(xwidth=0.01, yheight=0.01, zdepth=0.01, mosaic = 5,
    reflections="YBaCuO.lau")
107
108 A PG graphite crystal plate, cut for (002) reflections
109 Single_crystal(xwidth = 0.002, yheight = 0.1, zdepth = 0.1,
110 mosaic = 30, reflections = "C_graphite.lau",
111 ax=0,      ay=2.14,  az=-1.24,
112 bx = 0,    by = 0,   bz = 2.47,
113 cx = 6.71, cy = 0,   cz = 0)
114
115 A leucine protein, without multiple scattering
116 Single_crystal(xwidth=0.005, yheight=0.005, zdepth=0.005,
117 mosaic = 5, reflections="leucine.lau", order=1)
118
119 A Vanadium incoherent elastic scattering with multiple scattering
120 Single_crystal(xwidth=0.01, yheight=0.01, zdepth=0.01,
121 reflections="", sigma_abs=5.08, sigma_inc=4.935,
122 ax=3.0282, by=3.0282, cz=3.0282/2)
123
124 Also, always use a non-zero value of delta_d_d.
125
126 %VALIDATION:

```

```

127 This component has been validated.
128
129 This sample component can advantageously benefit from the SPLIT feature, e.
    g.
130 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). Use empty ("") or NULL for incoherent scattering only	0
geometry	str	Name of an Object File Format (OFF) or PLY file for complex geometry. The OFF/PLY file may be generated from XYZ coordinates using qhull/powercrust	0
mosaic_AB	arc_minutes, arc_minutes, 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 rotation 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 min- utes	Crystal mosaic (isotropic), gaussian RMS. Puts the crystal in the isotropic mosaic model state, thus disregarding other mosaicity parameters.	-1

Name	Unit	Description	Default
mosaic_a	arc min- utes	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 min- utes	Vertical (rotation around lattice vector b) mosaic (anisotropic), gaussian RMS.	-1
mosaic_c	arc min- utes	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 $\text{fm}^2$ . barns=1 for laz and isotropic constant elastic scattering (reflections=NULL), barns=0 for lau type files	0
ax	AA or AA <sup>-1</sup>	Coordinates of first (direct/recip) unit cell vector	0
ay		a on y axis	0
az		a on z axis	0
bx	AA or AA <sup>-1</sup>	Coordinates of second (direct/recip) unit cell vector	0
by		b on y axis	0
bz		b on z axis	0
cx	AA or AA <sup>-1</sup>	Coordinates of third (direct/recip) unit cell vector	0
cy		c on y axis	0
cz		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

Name	Unit	Description	Default
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
powder	1	Flag to indicate powder mode, for simulation of Debye-Scherrer cones via random crystallite orientation. A powder texture can be approximated with $0 < \text{powder} < 1$	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 $0 < \text{PG} < 1$ with main axis on 'c'	0

## Links

- Component source code found in file `Single_crystal.comp`.
- See <http://icsd.ill.fr> ICSD Inorganic Crystal Structure Database
- <http://www.ncnr.nist.gov/resources/n-lengths/> Cross sections for single elements
- <http://www.ncnr.nist.gov/resources/sldcalc.html> Cross sections for compounds
- <http://www.webelements.com/> Web Elements
- <http://www.ill.eu/sites/fullprof/index.html> Fullprof powder refinement
- <http://www.crystallographica.com/> Crystallographica software
- <http://www.geomview.org> Geomview and Object File Format (OFF)
- Java version of Geomview (display only) <http://www.holmes3d.net/graphics/roffview/> jroff.jar

- `<a href="http://qhull.org">qhull</a>`
- `<a href="http://www.cs.ucdavis.edu/~amenta/powercrust.html">powercrust</a>`



## 8.5. The Sans\_spheres McStas Component

Sample for Small Angle Neutron Scattering - hard spheres in thin solution, mono disperse.

### Identification

- **Site:**
- **Author:** P. Willendrup, K. Lefmann, L. Arleth
- **Origin:** Risoe
- **Date:** 19.12.2003

### Description

```
1 Sample for use in a SANS instrument, models hard, mono disperse spheres in
  thin solution.
2 The shape of the sample may be a filled box with dimensions
3 xwidth, yheight, zdepth, a cylinder with dimensions radius and yheight,
4 a filled sphere with radius.
5
6 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	AA	Radius of scattering hard spheres	100
Phi	1	Particle volume fraction	1e-3
Delta_rho	fm/AA <sup>3</sup>	Excess scattering length density	0.6
sigma_abs	m <sup>-1</sup>	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		0
target_y	m	position of target to focus at	0
target_z	m		6
target_index	1	Relative index of component to focus at, e.g. next is +1	0

Name	Unit	Description	Default
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
focus_r	m	Detector (disk-shaped) radius	0

## Links

- Component source code found in file `Sans_spheres.comp`.
- The test/example instrument [SANS.instr](../../examples/SANS.instr).

## 8.6. 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

- **Site:**
- **Author:** Kim Lefmann
- **Origin:** Risoe
- **Date:** 04.02.04

### Description

```
1 Single-cylinder shape.
2 Absorption included.
3 No multiple scattering.
4 No incoherent scattering emitted.
5 No attenuation from coherent scattering. No Bragg scattering.
6 fcc crystal n.n. interactions only
7 One phonon branch only -> phonon polarization not accounted for.
8 Bravais lattice only. (i.e. just one atom per unit cell)
9
10 Algorithm:
11 0. Always perform the scattering if possible (otherwise ABSORB)
12 1. Choose direction within a focusing solid angle
13 2. Calculate the zeros of  $(E_i - E_f - \hbar \omega(\kappa))$  as a function of  $k_f$ 
14 3. Choose one value of  $k_f$  (always at least one is possible!)
15 4. Perform the correct weight transformation
```

### Input parameters

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

Name	Unit	Description	Default
<b>radius</b>	m	Outer radius of sample in (x,z) plane	
<b>yheight</b>	m	Height of sample in y direction	
<b>sigma_abs</b>	barns	Absorption cross section at 2200 m/s per atom	
<b>sigma_inc</b>	barns	Incoherent scattering cross section per atom	
<b>a</b>	AA	fcc Lattice constant	
<b>b</b>	fm	Scattering length	
<b>M</b>	a.u.	Atomic mass	

Name	Unit	Description	Default
<b>c</b>	meV/Å <sup>-1</sup>	Velocity of sound	
<b>DW</b>	1	Debye-Waller factor	
<b>T</b>	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
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](../../../examples/Test_Phonon.instr)

## 8.7. The Isotropic\_Sqw McStas Component

Isotropic sample handling multiple scattering and absorption for a general  $S(q,w)$  (coherent and/or incoherent/self)

### Identification

- **Site:**
- **Author:** E. Farhi, V. Hugouvieux
- **Origin:** ILL
- **Date:** August 2003

### Description

```
1 An isotropic sample handling multiple scattering and including as input the
2 dynamic structure factor of the chosen sample (e.g. from Molecular
3 Dynamics). Handles elastic/inelastic, coherent and incoherent scattering -
4 depending on the input S(q,w) - with multiple scattering and absorption.
5 Only the norm of q is handled (not the vector), and thus suitable for
6 liquids, gases, amorphous and powder samples.
7
8 If incoherent/self S(q,w) file is specified as empty (0 or "") then the
9 scattering is constant isotropic (Vanadium like).
10 In case you only have one S(q,w) data containing both coherent and
11 incoherent contributions you should e.g. use 'Sqw_coh' and set 'sigma_coh'
12 to the total scattering cross section. Set sigma_coh and sigma_inc to -1 to
    deactivate.
13
14 The implementation will automatically normalise S(q,w) so that S(q) -> 1 at
15 large q (parameter norm=-1). Alternatively, the S(q,w) data will be
    multiplied
16 by 'norm' for positive values. Use norm=0 or 1 to use the raw data as input
    .
17
18 The material temperature can be defined in the S(q,w) data files (see below
    )
19 or set manually as parameter T. Setting T=-1 disables detailed balance.
20 Setting T=-2 attempts to guess the temperature from the input S(q,w) data
21 which must then be non-classical and extend on both energy sides (+/-).
22 To use the S(q,w) data as is, without temperature effect, set T=-1 and norm
    =1.
23
24 Both non symmetric (quantum) and classical S(q,w) data sets can be given by
    mean
25 of the 'classical' parameter (see below).
26
27 Additionally, for single order scattering (order=1), you may restrict the
28 vertical spreading of the scattering area using d_phi parameter.
29
```

30 An important option to enhance statistics is to set 'p\_interact' to, say,  
 31 30 percent (0.3) in order to force a fraction of the beam to scatter. This  
 32 will result on a larger number of scattered events, retaining intensity.  
 33  
 34 If you use **this** component **and** produce valuable scientific results, please  
 35 cite authors with references bellow (in [Links](#links)).  
 36 E. Farhi et al, J Comp Phys 228 (2009) 5251  
 37  
 38 <b>Sample shape:</b>  
 39 Sample shape may be a cylinder, a sphere, a box **or** any other shape  
 40 box/plate: xwidth x yheight x zdepth (thickness=0)  
 41 hollow box/plate: xwidth x yheight x zdepth **and** thickness>0  
 42 cylinder: radius x yheight (thickness=0)  
 43 hollow cylinder: radius x yheight **and** thickness>0  
 44 sphere: radius (yheight=0 thickness=0)  
 45 hollow sphere: radius **and** thickness>0 (yheight=0)  
 46 any shape: geometry=OFF file  
 47  
 48 The complex geometry option handles any closed non-convex polyhedra.  
 49 It computes the intersection points of the neutron ray with the object  
 50 transparently, so that it can be used like a regular sample object.  
 51 It supports the OFF, PLY **and** NOFF file format but **not** COFF (colored faces).  
 52 Such files may be generated from XYZ data **using**:  
 53 qhull < coordinates.xyz Qx Qv Tv o > geomview.off  
 54 **or**  
 55 powercrust coordinates.xyz  
 56 **and** viewed with geomview **or** java -jar jroff.jar (see below).  
 57 The **default** size of the object depends of the OFF file data, but its  
 58 bounding box may be resized **using** xwidth, yheight **and** zdepth.  
 59  
 60 <b>Concentric components:</b>  
 61 This component has the ability to contain other components when used in  
 62 hollow cylinder geometry (namely sample environment, e.g. cryostat **and**  
 63 furnace structure). Such component 'shells' should be split into input **and**  
 64 output side surrounding the 'inside' components. First part must then use  
 65 'concentric=1' flag to enter the inside part. The component itself must be  
 66 repeated to mark the end of the concentric zone. The number of concentric  
 67 shells **and** number of components inside is **not** limited.  
 68  
 69 COMPONENT S\_in = Isotropic\_Sqw(Sqw\_coh="Al.laz", concentric=1, ...)  
 70 AT (0,0,0) RELATIVE sample\_position  
 71  
 72 COMPONENT something\_inside ... *// e.g. the sample itself or other materials*  
 73  
 74 COMPONENT S\_out = COPY(S\_in)(concentric=0)  
 75 AT (0,0,0) RELATIVE sample\_position  
 76  
 77 <b>Sqw file format:</b>  
 78 File format **for** S(Q,w) (coherent **and** incoherent) should contain 3 numerical  
 79 blocks, defining q axis values (vector), then energy axis values (vector),  
 80 then a matrix with one line per q axis value, containing Sqw values **for**  
 81 each energy axis value. Comments (starting with '#') **and** non numerical  
 lines  
 82 are ignored **and** used to separate blocks. Sampling must be regular.

```

83 Some parameters can be specified in comment lines , namely (00 is a
    numerical value):
84
85 # sigma_abs    00 absorption scattering cross section in [barn]
86 # sigma_inc    00 coherent scattering cross section in [barn]
87 # sigma_coh    00 incoherent scattering cross section in [barn]
88 # Temperature  00 in [K]
89 # V_rho        00 atom density per Angs^3
90 # density      00 in [g/cm^3]
91 # weight       00 in [g/mol]
92 # classical    00 [0=contains Bose factor (measurement) ; 1=classical
    symmetric]
93
94 Example:
95 # q axis values
96 # vector of m values in Angstroem-1
97 0.001000 .... 3.591000
98 # w axis values
99 # vector of n values in meV
100 0.001391 ... 1.681391
101 # sqw values (one line per q axis value)
102 # matrix of S(q,w) values (m rows x n values), one line per q value ,
103 9.721422 10.599145 ... 0.000000
104 10.054191 11.025244 ... 0.000000
105 ...
106 0.000000 ... 3.860253
107
108 See for instance file He4_liq_coh.sqw. Such files may be obtained from e.g.
    INX,
109 Nathan, Lamp and IDA softwares , as well as Molecular Dynamics (nMoldyn).
110 When the provided S(q,w) data is obtained from the classical correlation
    function
111 G(r,t), which is real and symmetric in time, the 'classical=1' parameter
112 should be set in order to multiply the file data with exp(hw/2kT).
    Otherwise ,
113 the S(q,w) is NOT symmetrised (classical). If the S(q,w) data set includes
    both
114 negative and positive energy values, setting 'classical=-1' will attempt to
115 guess what type of S(q,w) it is. The temperature can also be determined
    this way.
116 In case you do not know if the data is classical or quantum, assume it is
    usually classical
117 at high temperatures, and quantum otherwise (T < typical mode excitations)
    .
118 The positive energy values correspond to Stokes processes , i.e. material
    gains
119 energy, and neutrons loose energy. The energy range is symmetrized to allow
    up
120 and down scattering , taking into account detailed balance exp(-hw/2kT).
121
122 You may also generate such S(q,w) 2D files using <a href="http://ifit.
    mccode.org/McStas.html#mozTocId297488">iFit </a>
123
124 <b>Powder file format:</b>

```

```

125 Files for coherent elastic powder scattering may also be used.
126 Format specification follows the same principle as in the PowderN
127 component, with parameters:
128
129 powder_format=
130 Crystallographica: { 4,5,7,0,0,0,0, 0,0 }
131 Fullprof:          { 4,0,8,0,0,5,0, 0,0 }
132 Undefined:         { 0,0,0,0,0,0,0, 0,0 }
133 Lazy:              { 17,6,0,0,0,0,0,13,0 }
134 qSq:                { -1,0,0,0,0,0,1, 0,0 } // special case for [q,Sq] table
135 or:                { j,d,F2,DW,Delta_d/d,1/2d,q,F, strain }
136
137 or column indexes (starting from 1) given as comments in the file header
138 (e.g. '#column_j 4'). Refer to the PowderN component for more details.
139 Delta_d/d and Debye-Waller factor may be specified for all lines with the
140 'powder_Dd' and 'powder_DW' parameters.
141 The reflection list should be ordered by decreasing d-spacing values.
142
143 Additionally a special [q,Sq] format is also defined with:
144 powder_format=qSq
145 for which column 1 is 'q' and column 2 is 'S(q)'.
146
147 <b>Examples:</b>
148 1- Vanadium-like incoherent elastic scattering
149 Isotropic_Sqw(radius=0.005, yheight=0.01, V_rho=1/13.827,
150 sigma_abs=5.08, sigma_inc=4.935, sigma_coh=0)
151
152 2- liq-4He parameters
153 Isotropic_Sqw(..., Sqw_coh="He4_liq_coh.sqw", T=10, p_interact=0.3)
154
155 3- powder sample
156 Isotropic_Sqw(..., Sqw_coh="Al.laz")
157
158 %BUGS:
159 When used in concentric mode, multiple bouncing scattering
160 (traversing the hollow part) is not taken into account.
161
162 %VALIDATION
163 For Vanadium incoherent scattering mode, V_sample, PowderN, Single_crystal
164 and Isotropic_Sqw produce equivalent results, even though the two later are
165 more accurate (geometry, multiple scattering). Isotropic_Sqw gives same
166 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}



Name	Unit	Description	Default
Sqw_coh	str	Name of the file containing the values of Q, w and S(Q,w) Coherent part; Q in Angs-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 Object File Format (OFF) or PLY file for complex geometry. The OFF/PLY 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
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

Name	Unit	Description	Default
rho	AA-3	Density of scattering elements (nb atoms/unit cell V <sub>0</sub> ).	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
powder_Vc	AA^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).
- <http://www.ncnr.nist.gov/resources/n-lengths/> Cross sections for single elements
- <http://www.ncnr.nist.gov/resources/sldcalc.html> Cross sections for compounds
- <http://www.webelements.com/> Web Elements
- <http://www.ill.eu/sites/fullprof/index.html> Fullprof powder refinement
- <http://www.crystallographica.com/> Crystallographica software
- Example data file <http://www.ncnr.nist.gov/resources/sldcalc.html> He4\_liq\_coh.sqw
- The <http://www.ncnr.nist.gov/resources/sldcalc.html> PowderN component.
- The test/example instrument <http://www.ncnr.nist.gov/resources/sldcalc.html> Test\_Isotropic\_Sqw.instr
- <http://www.geomview.org> Geomview and Object File Format (OFF)
- Java version of Geomview (display only) <http://www.holmes3d.net/graphics/roffview/> jroff.jar
- <http://qhull.org> qhull
- <http://www.cs.ucdavis.edu/~amenta/powercrust.html> powercrust

## 9. 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 ??.

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.

## 9.1. The TOF\_monitor McStas Component

Rectangular Time-of-flight monitor.

### Identification

- **Site:**
- **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`.

## 9.2. The TOF2E\_monitor McStas Component

TOF-sensitive monitor, converting to energy

### Identification

- **Site:**
- **Author:** Kim Lefmann and Helmuth Schoeber
- **Origin:** Risoe
- **Date:** Sept. 13, 2006

### Description

```
1 A square single monitor that measures the energy of the incoming neutrons
2 from their time-of-flight
3
4 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
xwidth	m	Width of detector. Overrides xmin, xmax	0
yheight	m	Height of detector. Overrides ymin, ymax	0
<b>Emin</b>	meV	Minimum energy to detect	
<b>Emax</b>	meV	Maximum energy to detect	
<b>T_zero</b>	s	Zero point in time	
<b>L_flight</b>	m	flight length user in conversion	

Name	Unit	Description	Default
restore_neutron	1	If set, the monitor does not influence the neutron state	0

## Links

- Component source code found in file `TOF2E_monitor.comp`.

## 9.3. The E\_monitor McStas Component

Energy-sensitive monitor.

### Identification

- **Site:**
- **Author:** Kristian Nielsen and Kim Lefmann
- **Origin:** Risoe
- **Date:** April 20, 1998

### Description

```

1 A square single monitor that measures the energy of the incoming neutrons.
2
3 Example: E_monitor(xmin=-0.1, xmax=0.1, ymin=-0.1, ymax=0.1,
4 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

Name	Unit	Description	Default
xwidth	m	Width of detector. Overrides xmin, xmax.	0
yheight	m	Height of detector. Overrides ymin, ymax.	0
<b>Emin</b>	meV	Minimum energy to detect	
<b>Emax</b>	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`.

## 9.4. The L\_monitor McStas Component

Wavelength-sensitive monitor.

### Identification

- **Site:**
- **Author:** Kristian Nielsen and Kim Lefmann
- **Origin:** Risoe
- **Date:** April 20, 1998

### Description

```

1 A square single monitor that measures the wavelength of the incoming
2 neutrons.
3
4 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



Name	Unit	Description	Default
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
<b>Lmin</b>	AA	Minimum wavelength to detect	
<b>Lmax</b>	AA	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`.

## 9.5. The PSD\_monitor McStas Component

Position-sensitive monitor.

### Identification

- **Site:**
- **Author:** Kim Lefmann
- **Origin:** Risoe
- **Date:** Feb 3, 1998

### Description

```

1 An (n times m) pixel PSD monitor. This component may also be used as a beam
2 detector.
3
4 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
<b>nx</b>	1	Number of pixel columns	90
<b>ny</b>	1	Number of pixel rows	90
<b>filename</b>	string	Name of file in which to store the detector image	0
<b>xmin</b>	m	Lower x bound of detector opening	-0.05
<b>xmax</b>	m	Upper x bound of detector opening	0.05
<b>ymin</b>	m	Lower y bound of detector opening	-0.05
<b>ymax</b>	m	Upper y bound of detector opening	0.05
<b>xwidth</b>	m	Width of detector. Overrides xmin, xmax	0
<b>yheight</b>	m	Height of detector. Overrides ymin, ymax	0
<b>restore_neutron</b>	1	If set, the monitor does not influence the neutron state	0
<b>nowritefile</b>	1	If set, monitor will skip writing to disk	0

## Links

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

## 9.6. The Divergence\_monitor McStas Component

Horizontal+vertical divergence monitor.

### Identification

- **Site:**
- **Author:** Kim Lefmann
- **Origin:** Risoe
- **Date:** Nov. 11, 1998

### Description

```
1 A divergence sensitive monitor. The counts are distributed in
2 (n times m) pixels.
3
4 Example: Divergence_monitor(nh=20, nv=20, filename="Output.pos",
```

```

5 xmin=-0.1, xmax=0.1, ymin=-0.1, ymax=0.1,
6 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
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		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		1

## Links

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

## 9.7. The DivPos\_monitor McStas Component

Divergence/position monitor (acceptance diagram).

### Identification

- Site:

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

## Description

```

1 2D detector for intensity as a function of position
2 and divergence, either horizontally or vertically (depending on the flag
   vertical).
3 This gives information similar to an acceptance diagram used
4 eg. to investigate beam profiles in neutron guides.
5
6 Example: DivPos_monitor(nh=20, ndiv=20, filename="Output.dip",
7 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	Vector definition of "forward" direction wrt. divergence, to be used e.g. when the monitor is rotated into the horizontal plane	0
ny	1		0
nz	1		1
vertical	1	Monitor intensity as a function of vertical divergence and position.	0

Name	Unit	Description	Default
nowritefile	1	If set, monitor will skip writing to disk	0

## Links

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

## 9.8. 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

- **Site:**
- **Author:** <mailto:farhi@ill.fr> Emmanuel Farhi
- **Origin:** <http://www.ill.fr> ILL
- **Date:** 14th Feb 2000.

### Description

```
1 This component is a general Monitor that can output 0/1/2D signals
2 It can produce many 1D signals (one for any variable specified in
3 option list), or a single 2D output (two variables correlation).
4 Also, an additional 'list' of neutron events can be produced.
5 By default, monitor is square (in x/y plane). A disk shape is also possible
6 The 'cylinder' and 'banana' option will change that for a banana shape
7 The 'sphere' option simulates spherical detector. The 'box' is a box.
8 The cylinder, sphere and banana should be centered on the scattering point.
9 The monitored flux may be per monitor unit area, and weighted by
10 a lambda/lambda(2200m/s) factor to obtain standard integrated capture flux.
11 In normal configuration, the Monitor_nD measures the current parameters
12 of the neutron that is being detected. USERVARS may be used in order
13 to study correlations between a neutron being detected in
14 a Monitor_nD place, and given parameters that are monitored elsewhere
15 (at the point of initialisation of the USERVARS).
16 The monitor can also act as a 3He gas detector, taking into account the
17 detection efficiency.
18
19 The 'bins' and 'limits' modifiers are to be used after each variable,
20 and 'auto', 'log' and 'abs' come before it. (eg: auto abs log hdiv bins=10
21 limits=[-5 5]) When placed after all variables, these two latter modifiers
22 apply to the signal (e.g. intensity). Unknown keywords are ignored.
23 If no limits are specified for a given observable, reasonable defaults will
24 be
25 applied. Note that these implicit limits are <b>even</b> applied in list
26 mode.
27
28 <b>Implicit limits for typical variables:</b>
29 (consult monitor_nd-lib.c if you don't find your variable here)
30 x, y, z: Derived from detection-object geometry
31 k: [0 10] Angs-1
32 v: [0 1e6] m/s
33 t: [0 1] s
34 p: [0 FLT_MAX] in intensity-units
```

```

33 vx, vy: [-1000 1000] m/s
34 vz: [0 10000] m/s
35 kx, ky: [-1 1] Angs-1
36 kz: [-10 10] Angs-1
37 energy, omega: [0 100] meV
38 lambda, wavelength: [0 100] Angs
39 sx, sy, sz: [-1 1] in polarisation-units
40 angle: [-50 50] deg
41 divergence, vdiv, hdiv, xdiv, ydiv: [-5 5] deg
42 longitude, latitude: [-180 180] deg
43 neutron: [0 simulaton_ncount]
44 id, pixel id: [0 FLT_MAX]
45 uservars u0,u1,u2,u3,u4,u5,u6,u7,u8,u9: [-1e10 1e10]
46
47 In the case of multiple components at the same position, the 'parallel'
48 keyword must be used in each instance instead of defining a GROUP.
49
50 <b>Possible options are</b>
51 Variables to record:
52 kx ky kz k wavevector [Angs-1] Wavevector on x,y,z and norm
53 vx vy vz v [m/s] Velocity on x,y,z and norm
54 x y z radius [m] Distance, Position and norm
55 xy, yz, xz [m] Radial position in xy, yz and xz plane
56 kxy kyz kxz [Angs-1] Radial wavevector in xy, yz and xz plane
57 vxy vyz vxz [m/s] Radial velocity in xy, yz and xz plane
58 t time [s] Time of Flight
59 energy omega [meV] energy of neutron
60 lambda wavelength [Angs] wavelength of neutron
61 sx sy sz [1] Spin
62 vdiv ydiv dy [deg] vertical divergence (y)
63 hdiv divergence xdiv [deg] horizontal divergence (x)
64 angle [deg] divergence from <z> direction
65 theta longitude [deg] longitude (x/z) for sphere and cylinder
66 phi latitude [deg] latitude (y/z) for sphere and cylinder
67
68 user0 user1 will monitor the [Mon_Name]_Vars.UserVariable
   {0|1|2|3|4|5}
69 user2 user3 to be assigned in an other component (see below)
70 user4 user5
71 user6 user7
72 user8 user9
73
74 Premonitoring: Please use uservars in place of the former
   PreMonitor_nD.
75
76 p intensity flux [n/s or n/cm^2/s]
77 ncounts n neutron [1] neutron ID, i.e current event index
78 pixel id [1] pixelID in histogram made of preceeding vars
   , e.g. 'theta y'. To set an offset PixelID use the 'min=value' keyword.
   Sets event mode.
79
80 <b>Other options keywords are:</b>
81 abs Will monitor the abs of the following variable or
   of the signal (if used after all variables)

```

```

82 auto                                     Automatically set detector limits for one/all
83 all {limits|bins|auto}                 To set all limits or bins values or auto mode
84 binary {float|double}                  with 'source' option, saves in compact files
85 bins=[bins=20]                          Number of bins in the detector along dimension
86 borders                                 To also count off-limits neutrons (X < min or X >
    max)
87 capture                                weight by lambda/lambda(2200m/s) capture flux
88 file=string                             Detector image file name. default is component
    name, plus date and variable extension.
89 incoming                               Monitor incoming beam in non flat det
90 limits=[min max]                        Lower/Upper limits for axes (see up for the
    variable unit)
91 list=[counts=1000] or all               For a long file of neutron characteristics with [
    counts] or all events
92 log                                     Will monitor the log of the following variable or
    of the signal (if used after all variables)
93 min=[min_value]                         Same as limits, but only sets the min or max
94 max=[max_value]
95 multiple                               Create multiple independant 1D monitors files
96 no or not                               Revert next option
97 outgoing                               Monitor outgoing beam (default)
98 parallel                               Use this option when the next component is at the
    same position (parallel components)
99 per cm2                                Intensity will be per cm^2 (detector area).
    Displays beam section.
100 per steradian                          Displays beam solid angle in steradian
101 signal=[var]                            Will monitor [var] instead of usual intensity
102 slit or absorb                          Absorb neutrons that are out detector
103 source                                  The monitor will save neutron states
104 inactivate                              To inactivate detector (0D detector)
105 verbose                                 To display additional informations
106 3He_pressure=[3 in bars]                The 3He gas pressure in detector. 3He_pressure=0
    is perfect detector (default)
107
108 Detector shape options (specified as xwidth,yheight,zdepth or x/y/z/min/max
    )
109 box                                     Box of size xwidth, yheight, zdepth.
110 cylinder                                To get a cylindrical monitor (diameter is xwidth
    or set radius, height is yheight).
111 banana                                  Same as cylinder, without top/bottom, on
    restricted angular area; use theta variable with limits to define arc.
    (diameter is xwidth or set radius, height is yheight).
112 disk                                    Disk flat xy monitor. diameter is xwidth.
113 sphere                                  To get a spherical monitor (e.g. a 4PI) (diameter
    is xwidth or set radius).
114 square                                  Square flat xy monitor (xwidth, yheight).
115 previous                               The monitor uses PREVIOUS component as detector
    surface. Or use 'geometry' parameter to specify any PLY/OFF geometry
    file.
116
117 <b>EXAMPLES:</b>
118 <ul>
119 <li>MyMon = Monitor_nD(xwidth = 0.1, yheight = 0.1, zdepth = 0,
120 &emsp;&emsp;options = "intensity per cm2 angle,limits=[-5 5] bins=10,with

```



```

121 &emsp;&emsp;borders, file = mon1");
122 will monitor neutron angle from [z] axis, between -5
123 and 5 degrees, in 10 bins, into "mon1.A" output 1D file
124
125 <li> options = "sphere theta phi outgoing"
126 for a sphere PSD detector (out beam) and saves into file "MyMon_[Date_ID].
    th_ph"
127
128 <li> options = "banana, theta limits=[10,130], bins=120, y"
129 a theta/height banana detector
130
131 <li> options = "angle radius all auto"
132 is a 2D monitor with automatic limits
133
134 <li> options = "list=1000 kx ky kz energy"
135 records 1000 neutron event in a file
136
137 <li> options = "multiple kx ky kz, auto abs log t, and list all neutrons"
138 makes 4 output 1D files and produces a complete list for all neutrons
139 and monitor log(abs(tof)) within automatic limits (for t)
140
141 <li> options = "theta y, sphere, pixel min=100"
142 a 4pi detector which outputs an event list with pixelID from the actual
143 detector surface, starting from index 100.
144
145 </ul>
146 To dynamically define a number of bins, or limits:
147 Use in DECLARE:      char op[256];
148 Use in INITIALIZE: sprintf(op, "lambda limits=[%g %g], bins=%i", lmin, lmax
    , lbin);
149 Use in TRACE:        Monitor_nD(... options=op ...)
150
151 <b>How to monitor any instrument/component variable into a Monitor_nD</b>
152 Suppose you want to monitor a variable 'age' which you assign somewhere in
153 the instrument:
154 COMPONENT MyMonitor = Monitor_nD(
155 xwidth = 0.1, yheight = 0.1,
156 user1="age", username1="Age of the Captain [years]",
157 options="user1, auto")
158 AT ...
159
160
161 %BUGS
162 The 'auto' option for guessing optimal variable bounds should NOT be used
    with MPI
163 as each process may use different limits.

```

## Input parameters

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

<b>Name</b>	<b>Unit</b>	<b>Description</b>	<b>Default</b>
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
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

Name	Unit	Description	Default
options	str	String that specifies the configuration of the monitor. The general syntax is "[x] options..." (see <b>Descr.</b>).	"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, 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 monitor)	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`.

## 10. 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. The components for doing this trick is **Virtual\_input** and **Virtual\_output**, which stores and reads neutron rays, respectively.

Other components perform the simulation of the instrument resolution functions. These are **Res\_sample** and **TOF\_Res\_sample**, which are to be placed at the sample position, and **Res\_monitor**, that should be localized at the position of the instrument detector.

**Progress\_bar** is a simulation utility that displays the simulation status, but assumes the form of a component.

## 10.1. **Virtual\_output**: Saving the first part of a split simulation

The component **Virtual\_output** stores the neutron ray parameters at the end of the first part of a split simulation. The idea is to let the next part of the split simulation be performed by another instrument file, which reads the stored neutron ray parameters by the component **Virtual\_input**.

All neutron ray parameters are saved to the output file, which is by default of “text” type, but can also assume the binary formats “float” or “double”. The storing of neutron rays continues until the specified number of simulations have been performed.

**buffer-size** may be used to limit the size of the output file, but absolute intensities are then likely to be wrong. Except when using MPI, we recommend to use the default value of zero, saving all neutron rays. The size of the file is then controlled indirectly with the general *ncounts* parameter.

## 10.2. **Virtual\_input**: Starting the second part of a split simulation

The component **Virtual\_input** resumes a split simulation where the first part has been performed by another instrument and the neutron ray parameters have been stored by the component **Virtual\_output**.

All neutron ray parameters are read from the input file, which is by default of “text” type, but can also assume the binary formats “float” and “double”. The reading of neutron rays continues until the specified number of rays have been simulated or till the file has been exhausted. If desirable, the input file can be reused a number of times, determined by the optional parameter “repeat-count”. This is only useful if the present simulation makes use of MC choices, otherwise the same outcome will result for each repetition of the simulation (see Appendix 2).

Care should be taken when dealing with absolute intensities, which will be correct only when the input file has been exhausted at least once.

The simulation ends with either the end of the repeated file counts, or with the normal end with *ncount* McStas simulation events. We recommend to control the simulation on **repeat-count** by using a very larger *ncount* value.

### 10.3. Res\_sample: A sample-like component for resolution calculation

The component **Res\_sample** scatters neutron rays isotropically in direction and uniformly in energy. Regardless of the state of the incoming neutron ray, all directions and energies for the scattered ray have the same probability, within specified intervals.

The component is meant for computation of the resolution function, but may also be used for test and debugging purposes. For actual calculations of the resolution function, **Res\_sample** should be used together with **Res\_monitor**, described in section 10.5.

The shape of **Res\_sample** is either a hollow cylinder or a rectangular box. The hollow cylinder shape is specified with the outer radius,  $r$  and thickness, respectively, and the height,  $h$ . If these parameters are unspecified, the shape is instead a box of dimensions  $x_w$ ,  $y_h$ , and  $z_d$ . The component only propagates neutron rays that are scattered; other rays are absorbed. The scattering probability is proportional to the neutron flight path length inside the sample, to make a true volume weighting of the sample. The reason for this is that the resolution function of an instrument is independent of any sample properties such as scattering and absorption cross sections but will in general depend on sample size and shape.

The point of scattering inside the sample is chosen uniformly along the neutron flight path inside the sample, and the scattered neutron ray is given a random energy and direction. This energy is selected in the interval  $[E_0 - \Delta E; E_0 + \Delta E]$  which hence must be chosen large enough to cover all interesting neutron energies. Similarly, the scattered direction is chosen in a user-specified range, either within a sphere of radius  $r_{\text{focus}}$ , within a rectangular target with measures  $(x_{\text{focus}}, y_{\text{focus}})$  or in the specified angular range. This target is positioned at the  $x_{\text{target}}, y_{\text{target}}, z_{\text{target}}$  point in space, or using the `target_index` for which e.g. 1 is the further component, -1 is the previous, etc...

A special feature, used when computing resolution functions, is that the component stores complete information about the scattering event in the output parameter `res_struct`. The information includes initial and final wave vectors, the coordinates of the scattering point, and the neutron weight after the scattering event. From this information the scattering parameters  $(\mathbf{Q}, \omega)$  can be recorded for every scattering event and used to compute the resolution function. For an example of using the information in the output parameter, see the description of the **Res\_monitor** component in section 10.5.

### 10.4. TOF\_Res\_sample: A sample-like component for TOF resolution calculation

The component **TOF\_Res\_sample** scatters neutron rays isotropically in position within a specified angular range. As for **Res\_sample**, this component is meant for computation of the resolution function, but in this case for one time bin in a time-of-flight (TOF) instrument. The component selects uniformly the neutron energy so that neutron arrival time at the TOF detector lies within one time bin, specified by  $t_0$  and  $\Delta t$ . For actual calculations of the resolution function, **TOF\_Res\_sample** should be

used together with **Res\_monitor**, described in section 10.5.

The shape of **TOF\_Res\_sample** is either a hollow cylinder or a rectangular box. The hollow cylinder shape is specified with the inner and outer radius,  $r_i$  and  $r_o$ , respectively, and the height,  $h$ . If these parameters are unspecified, the shape is instead a box of dimensions  $x_w$ ,  $y_h$ , and  $z_t$ .

The component only propagates neutron rays that are scattered; other rays are absorbed. As for **Res\_sample**, the scattering probability is proportional to the neutron flight path length inside the sample. The point of scattering in the sample is chosen uniformly along the neutron flight path inside the sample, and the scattered direction is chosen in a user-specified range, either within a sphere of radius  $r_{\text{foc}}$ , within a rectangular target with measures  $(x_{\text{focus}}, y_{\text{focus}})$  or in the specified angular range. This target is positioned at the  $x_{\text{target}}, y_{\text{target}}, z_{\text{target}}$  point in space, or using `target_index`.

This component stores complete information about the scattering event in the output parameter `res_struct`, see **Res\_Sample**.

## 10.5. Res\_monitor: The monitor for resolution calculation

The component **Res\_monitor** is used for calculating the resolution function of a particular instrument with detector of the given shape, size, and position. The shape of **Res\_monitor** is by default rectangular, but can be a box, a sphere, a disk, or a cylinder, depending on the parameter “options”. The component works like a normal monitor, but also records all scattering events and stores them to a file that can later be read by the McStas frontend tool `mcresplot`.

For time-of-flight (TOF) instruments, **Res\_monitor** should be understood as giving the resolution of one time bin of the TOF-detector only; the bin properties being specified in the preceding **TOF\_Res\_sample**.

As described in section 10.3, the **Res\_monitor** should be used in connection with one of the components **Res\_sample** or **TOF\_Res\_sample**, the name of which should be passed as an input parameter to **Res\_monitor**. For example

```
1 COMPONENT mysample = Res_sample( ... )
2 ...
3 COMPONENT det = Res_monitor(res_sample_comp = mysample, ...)
4 ...
```

The output file is in ASCII format, one line per scattering event, with the following columns:

- $\mathbf{k}_i$ , the three components of the initial wave vector.
- $\mathbf{k}_f$ , the three components of the final wave vector.
- $\mathbf{r}$ , the three components of the position of the scattering event in the sample.
- $p_i$ , the neutron weight just after the scattering event.

- $p_f$ , the relative neutron weight adjustment from sample to detector (so the total weight in the detector is  $p_i p_f$ ).

From  $\mathbf{k}_i$  and  $\mathbf{k}_f$ , we may compute the scattering parameters  $\kappa = \mathbf{k}_i - \mathbf{k}_f$  and  $\hbar\omega = \hbar^2/(2m_n)(\mathbf{k}_i^2 - \mathbf{k}_f^2)$ . The vectors are given in the local coordinate system of the resolution sample component. The wave vectors are in units of  $\text{\AA}^{-1}$ , the energy transfer in meV.

The output parameters from **Res\_monitor** are the three count numbers,  $Nsum$ ,  $psum$ , and  $p2sum$ , and the handle *file* of the output file.



## 10.6. Progress\_bar: Simulation progress and automatic saving

<b>Name:</b>	Progress_bar
<b>Author:</b>	System
<b>Input parameters</b>	percent, flag_save, profile
<b>Optional parameters</b>	
<b>Notes</b>	

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 beginning 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 it 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.

## 10.7. Beam\_spy: A beam analyzer

<b>Name:</b>	Beam_spy
<b>Author:</b>	System
<b>Input parameters</b>	
<b>Optional parameters</b>	
<b>Notes</b>	should overlap previous component

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**.

# A. Polarization in McStas

P. Christiansen (Risø)  
May 20, 2026

## A.1. Introduction

In the current release of McStas there are components with polarization capabilities. At the moment all such components should be understood as under development as the amount of testing and debugging of these components is small, and there are known problems.

Here, we shall report on what have been done so far.

We first describe the polarization vector and how it is related to the neutron wavefunction (section A.2) and then the physics of simple components that we need in McStas is reviewed (section A.3). In the last two sections the actual McStas polarization components are first described (section A.4) and a list of test instruments in McStas is given (section A.5).

We rely heavily on the books [Lov84; Wil88] for the 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  $\hat{\sigma}$  (vector of operators).

## A.2. 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:

$$\mathbf{P} = \frac{\langle \hat{\mathbf{S}} \rangle}{s}, \quad (\text{A.1})$$

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:

$$\mathbf{P} = \frac{\sum_i \mathbf{P}_i}{N}. \quad (\text{A.2})$$

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:

$$P = \frac{n_{\uparrow} - n_{\downarrow}}{n_{\uparrow} + n_{\downarrow}}, \quad (\text{A.3})$$

where  $\nu$  ( $n_{\downarrow}$ ) is the number of neutrons with spin up (down).

For a given neutron the probability of the neutron being spin up,  $P(\uparrow)$ , is:

$$P(\uparrow) = \frac{n_{\uparrow}}{n_{\uparrow} + n_{\downarrow}} = \frac{n_{\uparrow} + (n_{\downarrow} - n_{\downarrow})/2}{n_{\uparrow} + n_{\downarrow}} = \frac{1 + P}{2}, \quad (\text{A.4})$$

and  $P(\downarrow) = 1 - P(\uparrow) = (1 - P)/2$ .

The expectation value of the 'spin' operator,  $\hat{\sigma}$ , which can be expressed by the Pauli matrices, is the polarization vector  $\mathbf{P}$ ,  $\mathbf{P} = \langle \hat{\sigma} \rangle \equiv \langle \chi | \hat{\sigma} | \chi \rangle$ . The most general form of the spin wave-function  $\chi$  for a neutron (spin 1/2) is:

$$\chi = a\chi_{\uparrow} + b\chi_{\downarrow}, \quad (\text{A.5})$$

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:

$$P_x = \langle \chi | \hat{\sigma}_x | \chi \rangle = 2\text{Re}(a^*b) \quad (\text{A.6})$$

$$P_y = \langle \chi | \hat{\sigma}_y | \chi \rangle = 2\text{Im}(a^*b) \quad (\text{A.7})$$

$$P_z = \langle \chi | \hat{\sigma}_z | \chi \rangle = |a|^2 - |b|^2 \quad (\text{A.8})$$

This shows the relation of the polarization vector to the neutron wave function.

The neutron magnetic moment operator can be expressed in terms of  $\hat{\sigma}$ , as:

$$\hat{\mu}_{\mathbf{n}} = \mu_n \hat{\sigma}, \quad (\text{A.9})$$

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:

$$P(\uparrow | \mathbf{n}) = \frac{1 + \mathbf{n} \cdot \mathbf{S}}{2}. \quad (\text{A.10})$$

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:

$$F_N(\boldsymbol{\kappa}) = \sum_{\mathbf{d}} \exp(i\boldsymbol{\kappa} \cdot \mathbf{d}) \bar{b}_d, \quad (\text{A.11})$$

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:

$$F_M(\boldsymbol{\kappa}) = \gamma_n r_0 \sum_{\mathbf{d}} \exp(i\boldsymbol{\kappa} \cdot \mathbf{d}) \frac{1}{2} g_d F_d(\boldsymbol{\kappa}) \langle \hat{S}_d \rangle \sigma_d, \quad (\text{A.12})$$

where  $r_0 = \frac{\mu_0}{4\pi} \frac{e^2}{m_e} = 2.818 \times 10^{-15} \text{m}$ ,  $g = 2$  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.2.1. 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) <sup>1</sup>

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:

$$\frac{1}{\gamma_n} \frac{d\boldsymbol{\mu}}{dt} = \boldsymbol{\mu} \times \mathbf{B} \quad (\text{A.13})$$

The magnetic moment  $\boldsymbol{\mu}$  can be related to the polarization as  $\boldsymbol{\mu} = \gamma_n \mathbf{P}/2$ , and inserting in Eq. A.13 we find:

$$\frac{d\mathbf{P}}{dt} = \gamma_n \mathbf{P} \times \mathbf{B} \quad (\text{A.14})$$

---

<sup>1</sup>Note that if we had used  $S$  (with values  $S = \pm 1$ ) to define  $\gamma_n$  we would get  $\gamma_n = -1.913$  which is also commonly used.

In the simple case where  $\mathbf{B} = (0, 0, B)$ , we find the solution ([Wil88] p. 18) :

$$\begin{aligned} P_X(t) &= \cos(\omega_L t) P_X(0) - \sin(\omega_L t) P_Y(0) \\ P_Y(t) &= \sin(\omega_L t) P_X(0) + \cos(\omega_L t) P_Y(0) \\ P_Z(t) &= P_Z(0), \end{aligned} \quad (\text{A.15})$$

where  $\omega_L = -\gamma_n B/\hbar$  is the Larmor frequency.

The equations above were checked against the equations in the “polarimetric 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.

### A.3. 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:

$$\hat{v} = \hat{\beta} + \hat{\alpha} \cdot \hat{\sigma} \quad (\text{A.16})$$

The first term does not affect the spin, while the second term can change the spin. Let us just remind here that:

$$\begin{aligned} \hat{\sigma}_x \chi_\uparrow &= \chi_\downarrow, & \hat{\sigma}_y \chi_\uparrow &= i\chi_\downarrow, & \hat{\sigma}_z \chi_\uparrow &= \chi_\uparrow, \\ \hat{\sigma}_x \chi_\downarrow &= \chi_\uparrow, & \hat{\sigma}_y \chi_\downarrow &= -i\chi_\uparrow, & \hat{\sigma}_z \chi_\downarrow &= -\chi_\downarrow. \end{aligned} \quad (\text{A.17})$$

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:

$$\hat{\rho} = \chi \chi^\dagger = \begin{pmatrix} |a|^2 & ab^* \\ ba^* & |b|^2 \end{pmatrix} = \frac{1}{2}(\mathcal{I} + \mathbf{P} \cdot \hat{\sigma}), \quad (\text{A.18})$$

where  $\chi$  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):

$$\frac{d\sigma}{d\Omega} = \text{Tr} \hat{\rho} \hat{v}^\dagger \hat{v} = \sum_{\lambda, \lambda'} p_\lambda \text{Tr} \hat{\rho} \langle \lambda | \hat{V}^\dagger(\boldsymbol{\kappa}) | \lambda' \rangle \langle \lambda' | \hat{V}(\boldsymbol{\kappa}) | \lambda \rangle \delta(E_\lambda - E_{\lambda'}), \quad (\text{A.19})$$

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:

$$\mathbf{P}' \frac{d\sigma}{d\Omega} = \text{Tr} \hat{\rho} \hat{v}^\dagger \hat{\sigma} \hat{v} = \sum_{\lambda, \lambda'} p_\lambda \text{Tr} \hat{\rho} \langle \lambda | \hat{V}^\dagger(\boldsymbol{\kappa}) | \lambda' \rangle \hat{\sigma} \langle \lambda' | \hat{V}(\boldsymbol{\kappa}) | \lambda \rangle \delta(E_\lambda - E_{\lambda'}) \quad (\text{A.20})$$

Inserting Eq. A.16 in Eq. A.19 and Eq. A.20 results in the two master equations for polarized neutron scattering:

$$\text{Tr} \hat{\rho} \hat{v}^\dagger \hat{v} = \hat{\alpha}^\dagger \cdot \hat{\alpha} + \hat{\beta}^\dagger \hat{\beta} + \hat{\beta}^\dagger (\hat{\alpha} \cdot \mathbf{P}) + (\hat{\alpha}^\dagger \cdot \mathbf{P}) \hat{\beta} + i \mathbf{P} \cdot (\hat{\alpha}^\dagger \times \hat{\alpha}), \quad (\text{A.21})$$

and

$$\text{Tr} \hat{\rho} \hat{v}^\dagger \hat{\sigma} \hat{v} = \hat{\beta}^\dagger \hat{\alpha} + \hat{\alpha}^\dagger \hat{\beta} + \hat{\beta}^\dagger \hat{\beta} \mathbf{P} + \hat{\alpha}^\dagger (\hat{\alpha} \cdot \mathbf{P}) + (\hat{\alpha}^\dagger \cdot \mathbf{P}) \hat{\alpha} - \mathbf{P} (\hat{\alpha}^\dagger \cdot \hat{\alpha}) - i \hat{\alpha}^\dagger \times \hat{\alpha} + i \hat{\beta}^\dagger (\hat{\alpha} \times \mathbf{P}) + i (\mathbf{P} \times \hat{\alpha}^\dagger) \hat{\beta}. \quad (\text{A.22})$$

Based on these two equations and the interaction potentials all the results presented in the following are derived in [Lov84].

### A.3.1. Example: Nuclear scattering

The nuclear scattering potential for a crystal is:

$$\hat{V}_N(\boldsymbol{\kappa}) = \sum_{\mathbf{l}, \mathbf{d}} \exp(i \boldsymbol{\kappa} \cdot \mathbf{R}_{l\mathbf{d}}) (A_{l\mathbf{d}} + \frac{1}{2} B_{l\mathbf{d}} \hat{\sigma} \cdot \hat{\mathbf{l}}_{l\mathbf{d}}), \quad (\text{A.23})$$

so that

$$\hat{\alpha} = \sum_{\mathbf{l}, \mathbf{d}} \exp(i \boldsymbol{\kappa} \cdot \mathbf{R}_{l\mathbf{d}}) \frac{1}{2} B_{l\mathbf{d}} \hat{\mathbf{l}}_{l\mathbf{d}} \quad (\text{A.24})$$

$$\hat{\beta} = \sum_{\mathbf{l}, \mathbf{d}} \exp(i \boldsymbol{\kappa} \cdot \mathbf{R}_{l\mathbf{d}}) A_{l\mathbf{d}}, \quad (\text{A.25})$$

where  $\hat{\mathbf{l}}$  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  $\hat{\alpha}$  (three last terms in Eq. A.21) disappears. The scattering cross section ends up being (see [Lov84] p. 159):

$$\frac{d\sigma}{d\Omega} = \sum_{\mathbf{l}, \mathbf{d}, \mathbf{l}', \mathbf{d}'} \exp(i \boldsymbol{\kappa} \cdot (\mathbf{R}_{l\mathbf{d}} - \mathbf{R}_{l'\mathbf{d}'})) (|\overline{A_d}|^2 + \delta_{\mathbf{l}, \mathbf{l}'} \delta_{\mathbf{d}, \mathbf{d}'} [|\overline{A_d}|^2 - |\overline{A_d}|^2 + \frac{1}{4} |\overline{B_d}|^2 I_d(I_d+1)]) \quad (\text{A.26})$$

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:

$$\mathbf{P}' \frac{d\sigma}{d\Omega} = \sum_{\mathbf{l}, \mathbf{d}, \mathbf{l}', \mathbf{d}'} \exp(i\mathbf{\kappa} \cdot (\mathbf{R}_{ld} - \mathbf{R}_{l'd'})) \mathbf{P}' (|\overline{A_d}|^2 + \delta_{\mathbf{l}, \mathbf{l}'} \delta_{\mathbf{d}, \mathbf{d}'} [|\overline{A_d}|^2 - |\overline{A_d}|^2 - \frac{1}{12} |\overline{B_d}|^2 I_d(I_d+1)]) \quad (\text{A.27})$$

Comparing Eq. A.26 and Eq. A.27 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.3.2. Example: Polarizing Monochromator and Guides

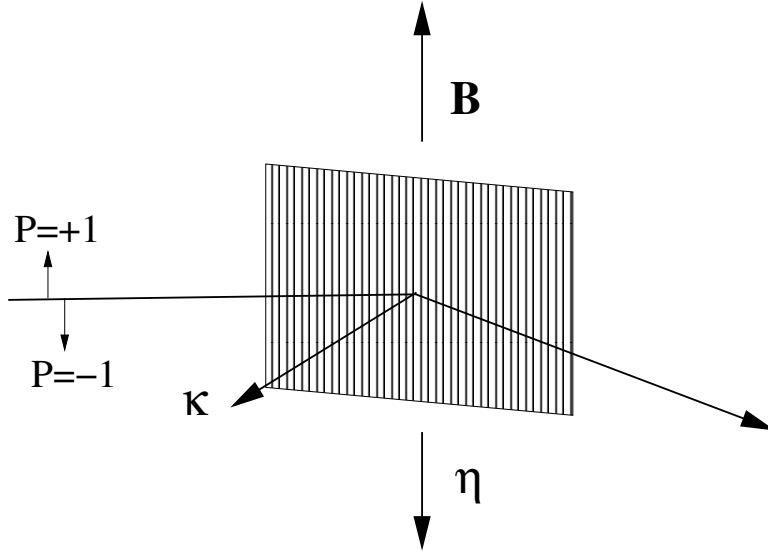


Figure A.1.: Principle and geometry of a polarizing monochromator.

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):

$$d\sigma/d\Omega = F_N(\boldsymbol{\kappa})^2 + 2F_N(\boldsymbol{\kappa})F_M(\boldsymbol{\kappa})(\mathbf{P} \cdot \tilde{\boldsymbol{\eta}}) + F_M(\boldsymbol{\kappa})^2 \quad (\text{A.28})$$

$$\mathbf{P}' d\sigma/d\Omega = \mathbf{P}[F_N(\boldsymbol{\kappa})^2 - F_M(\boldsymbol{\kappa})^2] + \tilde{\boldsymbol{\eta}}[2F_N(\boldsymbol{\kappa})F_M(\boldsymbol{\kappa}) + 2(\tilde{\boldsymbol{\eta}} \cdot \mathbf{P})F_M(\boldsymbol{\kappa})^2] \quad (\text{A.29})$$

NB! Note that in [Wil88] Eq. 2.2.25 there is a minus in front of the second term in Eq. A.28. 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:

$$\mathbf{P}' = \frac{\tilde{\boldsymbol{\eta}} 2F_N(\boldsymbol{\kappa})F_M(\boldsymbol{\kappa})}{d\sigma/d\Omega} = \frac{2F_N(\boldsymbol{\kappa})F_M(\boldsymbol{\kappa})}{F_N(\boldsymbol{\kappa})^2 + F_M(\boldsymbol{\kappa})^2} \tilde{\boldsymbol{\eta}}, \quad (\text{A.30})$$

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.28 to determine  $F_N(\boldsymbol{\kappa})$  and  $F_M(\boldsymbol{\kappa})$ :

$$R_\uparrow = k(F_N(\boldsymbol{\kappa}) + F_M(\boldsymbol{\kappa}))^2, \quad (\text{A.31})$$

$$R_\downarrow = k(F_N(\boldsymbol{\kappa}) - F_M(\boldsymbol{\kappa}))^2. \quad (\text{A.32})$$

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:

$$F_N(\boldsymbol{\kappa}) = \frac{\sqrt{R_\uparrow} + \sqrt{R_\downarrow}}{2}, \quad (\text{A.33})$$

$$F_M(\boldsymbol{\kappa}) = \frac{\sqrt{R_\uparrow} - \sqrt{R_\downarrow}}{2}. \quad (\text{A.34})$$

When  $F_N(\boldsymbol{\kappa})$  and  $F_M(\boldsymbol{\kappa})$  are determined from these equations, Eq. A.28 and Eq. A.29 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  $\tilde{\eta}^2(\tilde{\eta} \cdot \mathbf{P})F_M(\boldsymbol{\kappa})^2$  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.28 and Eq. A.29 without the simplifying geometrical constraints for monochromators and guides.

## A.4. New McStas Components

The components written so far can be divided into four groups:

- **Polarizers:** Components used to make the beam polarized.
- **Monitors:** Unphysical detectors that can measure the polarization of the neutrons.
- **Magnetic fields:** Components used to handle magnetic fields.
- **Samples:** Samples that affects the polarization.

### A.4.1. Polarizers

Some of the most common ways of polarizing a beam have been implemented.

- **Set\_pol:** This unphysical component can be used in two ways. Either to hard code the polarization to the vector  $(px, py, pz)$  or when `randomOn!=0` to set the polarization vector to a random vector on the unit sphere.
- **Monochromator\_pol:** A monochromator that only does the  $n = 1$  reflection. For each neutron it calculates the wavelength which would give Bragg reflection,  $\lambda_{\text{Bragg}}$ , and it then calculates, based on one mosaicity and one d-spread, the reflection probability given the neutrons actual  $\lambda$ . The reflection probability is a

Gaussian in  $\Delta\lambda = \lambda - \lambda_{\text{Bragg}}$ , with the peak reflectivity and polarization calculated as described in section A.3.2.

NB! Note that this monochromator reflects the neutrons billiard-like. In **Monochromator\_flat** the mosaicity of the reflecting crystal is taken into account, but the d-spread is not taken into account. One should implement d-spread and mosaicity in a way similar to what is done in **Single crystal**.

- **Pol\_mirror:** Plane with a reflection probability for up and down. There are 3 options: always reflect, always transmit, or random select transmit/reflect.

NB! Note that at the moment the plane only reflects from one side (because it uses PROP\_Z0).

- **Pol\_bender:** Curved guide with the possibilities to insert multiple slits, and have the end gap parallel to the entrance or following the guide angle. It is possible to select different coatings (mirror parameters) for each of the four sides.
- **Pol\_guide\_vmirror:** Straight guide with non-polarizing coatings with two polarizing super mirrors sitting in a V shape inside.

Note that for all the polarizing guides it is possible to define analytical functions or use tables for the up and down reflectivity descriptions.

#### A.4.2. Detectors

- **Pol\_monitor:** One defines a vector  $\mathbf{m} = (mx, my, mz)$  for the monitor and measures the projection of the spin along this vector i.e.  $\mathbf{m} \cdot \mathbf{S}$ .
- **PolLambda\_monitor:** Measures the projection of the spin along the defined vector  $\mathbf{m}$  (see **Pol\_monitor**) as a function of the wavelength  $\lambda$ .
- **MeanPolLambda\_monitor:** Measures the *average* projection of the spin along the defined vector  $\mathbf{m}$  (see **Pol\_monitor**) as a function of the wavelength  $\lambda$ .

NB! currently the error on the mean is shown ( $\sigma/\sqrt{(N)}$ ), but it might make more sense to show the spread ( $\sigma$ ).

#### A.4.3. Magnetic fields

Much inspiration for the components and the tests have been found in [SD01].

- **Pol\_constBfield:** A rectangular box with a constant magnetic field in the y-direction. The x- and z-components of the spin precess with the Larmor frequency  $\omega_L$ . It is possible to define the field in terms of a wavelength so that the spin will precess 180 degrees for the given wavelength. The component can be rotated to have the field along another axis.
- **Pol\_simpleBfield:** The first attempt at a component for handling general magnetic fields. It is a concentric component where you define a start and stop component for each field, but this allows other components, e.g. monitors, to be put inside the field. The component overloads the propagation routines so that numerical spin propagation is done for analytical magnetic fields.

NB! At the moment both components does not really check the boundaries of the field on the sides, but merely assumes that the field starts at the entrance plane and stops at the exit plane.

Also, some optimization remains for the numerical component and it would be nice to support tabulated magnetic field files. However, the framework developed for **Pol\_simpleBfield** is very general and should easily facilitate these changes.

#### A.4.4. Samples

- **V\_sample:** Modified the sample so that the scattered neutron has  $\mathbf{P}' = -1/3\mathbf{P}$ . Note that this component does not handle multiple scattering, so this approach is correct. If the components handled multiple scattering the polarization should be set to  $\mathbf{P}' = (-1/3)^n\mathbf{P}$ , where  $n$  is the number of scatterings.

### A.5. Tests With New Components

All the test instruments can be found in the McStas examples folder (go to “Neutron site/tests” in mcgui).

There are basically two kind of tests. The first kind of tests shows that the polarizing component can reproduce the same results as a similar non-polarizing component:

- *Test\_Monochromators.instr* : Intercomparison of **Monochromator\_flat** and **Monochromator\_pol**.
- *Test\_Pol\_Bender\_Vs\_Guide\_Curved.instr* : Intercomparison of **Guide\_curved** and **Pol\_bender**.

The second type of test illustrates the polarizing capabilities of the component:

- *Test\_Magnetic\_Constant.instr* : Constant magnetic field.

- *Test\_Magnetic\_Majorana.instr* : Linearly decreasing field with small transverse component.
- *Test\_Magnetic\_Rotation.instr* : Rotating magnetic field.
- *Test\_Magnetic\_Userdefined.instr* : Example of how to make a user defined analytic magnetic field that can also depend on time.
- *Test\_Pol\_Bender.instr* : Illustrates beam polarization with the **Pol\_bender**.
- *Test\_Pol\_Set.instr* : Tests **Pol\_set**.
- *Test\_Pol\_Guide\_Vmirror.instr* : Illustrates beam polarization with the **Pol\_guide\_vmirror**.
- *Test\_Pol\_Mirror.instr* : Illustrates beam polarization with the **Pol\_mirror**.
- *Test\_Pol\_TripleAxis.instr* : An example of a triple axis spectrometer with polarizing monochromators, a vanadium sample, and a spin flipper.

## B. 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.

### B.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.

#### B.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.
- **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).

### B.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 matrix 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 useful 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.

### B.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*  $t, \phi_x, \phi_y, \phi_z$ ) Get transformation matrix for rotation first  $\phi_x$  around x axis, then  $\phi_y$  around y, and last  $\phi_z$  around z.  $t$  should be a 'Rotation' ([3][3] 'double' matrix).
- **rot\_mul**(*Rotation*  $t1, \text{Rotation } t2, \text{Rotation } t3$ ) performs  $t3 = t1.t2$ .
- **rot\_copy**(*Rotation*  $dest, \text{Rotation } src$ ) performs  $dest = src$  for Rotation arrays.
- **rot\_transpose**(*Rotation*  $src, \text{Rotation } dest$ ) performs  $dest = src^t$ .
- **rot\_apply**(*Rotation*  $t, \text{Coords } a$ ) returns a Coord structure which is  $t.a$

### B.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  $\varphi$  (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**( $\&t1, \&t2, A, B, C$ ). Solves the  $2^{nd}$  order equation  $At^2 + Bt + C = 0$  and returns the solutions into pointers  $*t1$  and  $*t2$ . if  $t2$  is specified as NULL, only the smallest positive solution is returned in  $t1$ .

(\* 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.)



### B.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-dimensional detector. Integrated intensities error etc. is also reported as for DETECTOR\_OUT\_0D.
- **DETECTOR\_OUT\_3D(...)**. Used to output the results from a three-dimensional detector. Arguments are the same as in DETECTOR\_OUT\_2D, but with an additional  $z$  axis. Resulting data files are treated as 2D data, but the 3rd dimension is specified in the *type* field. 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.

### B.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  $r$ .

### B.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**(& $v_x$ , & $v_y$ , & $v_z$ , & $d\Omega$ , aim $_x$ , aim $_y$ , 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**(& $v_x$ , & $v_y$ , & $v_z$ , & $d\Omega$ , aim $_x$ , aim $_y$ , 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.

## B.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):

```

1 /* t_Table structure (most important members) */
2 double *data;      /* Use Table_Index(Table, i j) to get element [i,j] */
3 long    rows;      /* number of rows */
4 long    columns;   /* number of columns */
5 char    *header;   /* the header with comments */
6 char    *filename; /* file name or title */
7 double  min_x;     /* minimum value of 1st column/vector */
8 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 encountered (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 beginning 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 Beryllium filter) looks like:

```

1  # BeO transmission , as measured on IN12
2  # Thickness: 0.05 [m]
3  # [ k(Angs-1) Transmission (0-1) ]
4  # wavevector multiply
5  1.0500  0.74441
6  1.0750  0.76727
7  1.1000  0.80680
8  ...

```

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:

```

1  t_Table Table;           // declare a t_Table structure
2  char file []="BeO.trm";  // a file name
3  double x,y;
4
5  Table_Read(&Table, file, 1); // initialize and read the first numerical
   block
6  Table_Info(Table);        // display table informations
7  ...
8  x = Table_Index(Table, 2,5); // read the 3rd row, 6th column element
9                               // of the table. Indexes start at zero in C
10
11 y = Table_Value(Table, 1.45,1); // look for value 1.45 in 1st column (x
   axis)
12                               // and extract 2nd column value of that row
13 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:

```

1  t_Table *Table;         // declare a t_Table structure array
2  long n;

```

```

3  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
10 Table_Free_Array(Table); // ONLY use Table[i] with i < n !
      // free allocated memory for Table

```

You may look into, for instance, the source files for **Monochromator\_curved** or **Virtual\_input** for other implementation examples.

### B.3. Monitor\_nD Library

This library gathers a few functions used by a set of monitors e.g. **Monitor\_nD**, **Res\_monitor**, **Virtual\_output**, etc. 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.

### B.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.

### B.5. Vitess import/export Library

This library is used by the components **Vitess\_input** and **Vitess\_output**, as well as the **mcstas2vitess** utility. Refer to these components for implementation examples, and ask the authors for more details.

### B.6. Constants for unit conversion etc.

The following predefined constants are useful for conversion between units

Name	Value	Conversion from	Conversion to
<b>DEG2RAD</b>	$2\pi/360$	Degrees	Radians
<b>RAD2DEG</b>	$360/(2\pi)$	Radians	Degrees
<b>MIN2RAD</b>	$2\pi/(360 \cdot 60)$	Minutes of arc	Radians
<b>RAD2MIN</b>	$(360 \cdot 60)/(2\pi)$	Radians	Minutes of arc
<b>V2K</b>	$10^{10} \cdot m_N/\hbar$	Velocity (m/s)	<b>k</b> -vector ( $\text{\AA}^{-1}$ )
<b>K2V</b>	$10^{-10} \cdot \hbar/m_N$	<b>k</b> -vector ( $\text{\AA}^{-1}$ )	Velocity (m/s)
<b>VS2E</b>	$m_N/(2e)$	Velocity squared ( $\text{m}^2 \text{s}^{-2}$ )	Neutron energy (meV)
<b>SE2V</b>	$\sqrt{2e/m_N}$	Square root of neutron energy ( $\text{meV}^{1/2}$ )	Velocity (m/s)
<b>FWHM2RMS</b>	$1/\sqrt{8 \log(2)}$	Full width half maximum	Root mean square (standard deviation)
<b>RMS2FWHM</b>	$\sqrt{8 \log(2)}$	Root mean square (standard deviation)	Full width half maximum
<b>MNEUTRON</b>	$1.67492 \cdot 10^{-27} \text{ kg}$	Neutron mass, $m_n$	
<b>HBAR</b>	$1.05459 \cdot 10^{-34} \text{ Js}$	Planck constant, $\hbar$	
<b>PI</b>	3.14159265...	$\pi$	
<b>FLT_MAX</b>	3.40282347E+38F	a big float value	

## C. 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

- [Mcs] See <https://www.mcstas.org> (cit. on pp. 8, 15).
- [Git] See <https://github.com/mccode-dev/McCode/issues> (cit. on pp. 8, 15).
- [Nmi] See <http://neutron-eu.net/en> (cit. on p. 8).
- [Mcna] See <http://mcnsi.risoe.dk> (cit. on p. 8).
- [Ts2] See <http://ts-2.isis.rl.ac.uk> (cit. on p. 8).
- [Ess] See <http://www.ess-europe.de> (cit. on p. 8).
- [Sin] See <http://sine2020.eu> (cit. on p. 8).
- [Ics] ICSD, Inorganic Crystal Structure Database. See <http://icsd.ill.fr> (cit. on pp. 13, 14).
- [Tri] See <http://www.nea.fr/html/dbprog/tripoli-abs.html> (cit. on p. 41).
- [Mcnb] See <http://mcnp.lanl.gov> and <http://mcnp.lanl.gov> (cit. on p. 41).
- [Vit] See <http://www.hmi.de/projects/ess/vitess> (cit. on p. 41).
- [Bac75] G.E. Bacon. *Neutron Diffraction*. Oxford University Press, 1975 (cit. on p. 68).
- [Cop93] J.R.D. Copley. In: *J. Neut. Research* 1 (1993), p. 21 (cit. on p. 16).
- [DL03] A.-J. Dianoux and G. Lander. *ILL Neutron Data Booklet*. OCP Science, 2003 (cit. on pp. 13, 14).
- [Far+02] E. Farhi et al. In: *Appl. Phys.* A 74 (2002), S1471 (cit. on p. 16).
- [GRR92] Grimmett, G. R., and Stirzaker and D. R. *Probability and Random Processes, 2nd Edition*. Clarendon Press, Oxford, 1992 (cit. on p. 16).
- [Jam80] F. James. In: *Rep. Prog. Phys.* 43 (1980), p. 1145 (cit. on pp. 16, 20).
- [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. 8).
- [Lie05] K. Lieutenant. In: *J. Phys.: Condens. Matter* 17 (2005), S167 (cit. on p. 16).
- [Lov84] S.W. Lovesey. *Theory of neutron scattering from condensed matter*. Oxford Clarendon Press, 1984 (cit. on pp. 122, 125, 126, 128).
- [Pre+86] W. H. Press et al. *Numerical Recipes in C*. Cambridge University Press, 1986 (cit. on p. 138).
- [Sch+04] C. Schanzer et al. In: *Nucl. Instr. Meth.* A 529 (2004), p. 63 (cit. on p. 16).



- [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 pp. 123, 130).
- [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. 8).
- [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. 8, 15).
- [Wil88] W. Gavin Williams. *Polarized Neutrons*. Oxford Clarendon Press, 1988 (cit. on pp. 122, 125, 128).
- [ZLa04] G. Zsigmond, K. Lieutenant, and S. Manoshin et al. In: *Nucl. Instr. Meth.* A 529 (2004), p. 218 (cit. on p. 16).

# Index

- `%include` (McStas keyword), 133
- `ABSORB` (C macro, `mcstas-r.h`), 133
- `adapt_tree-lib` (library), **141**
- Adaptive sampling, 19
- `ALLOW_BACKPROP` (C macro, `mcstas-r.h`), 134
- Arm, 143
- `box_intersect` (C function, `mccode-r.c`), 137
- Bugs, 15
- C functions, 133
- Component, 143
  - instance, 143
- Constants, **141**
- Conversion
  - of units, 141
- Coordinate
  - retrieval functions, 134
  - system, 11
- `coords_add` (C function, `mccode-r.c`), 136
- `coords_get` (C function, `mccode-r.c`), 136
- `coords_set` (C function, `mccode-r.c`), 136
- `cylinder_intersect` (C function, `mccode-r.c`), 137
- Data files, 11
- Definition parameter, 143
- `DEG2RAD` (constant), **142**
- `DETECTOR_OUT_OD` (C macro, `mccode-r.h`), 137
- `DETECTOR_OUT_1D` (C macro, `mccode-r.h`), 137
- `DETECTOR_OUT_2D` (C macro, `mccode-r.h`), 137
- `DETECTOR_OUT_3D` (C macro, `mccode-r.h`), 137
- Direction focusing, 19
- Environment variable
  - `BROWSER`, 12
  - `MCSTAS`, 12
- Environment variables
  - `MCSTAS`, 133, 140
- Error estimate, 17
- `EXTEND` (McStas keyword), 134
- `FLT_MAX` (constant), **142**
- Focusing
  - importance sampling, 19
- `FWHM2RMS` (constant), **142**
- `GROUP` (McStas keyword), 134
- `HBAR` (constant), **142**
- Importance sampling
  - direction focusing, 19
- `INDEX_CURRENT_COMP` (C macro, `mccode-r.h`), 135
- Input parameter, 143
- `inside_rectangle` (C function, `mcstas-r.c`), 137
- Instrument, 143
- `K2V` (constant), **142**
- Kernel, 143
- Keyword
  - `EXTEND`, 12, 22
- Library, **133**
  - `adapt_tree-lib`, **141**
  - Components
    - data, 12–14
    - misc, 116
    - monitors, 100
    - optics, 42, 51
    - samples, 67
    - sources, 22
  - components
    - data, 140
    - share, 133
  - `mccode-r`, **133**
  - `mcstas-r`, **133**

- monitor\_nd-lib, **141**
- read\_table-lib, **138**
- read\_table-lib (Read\_Table), 11
- run-time, **133**
- vitess-lib, **141**
  
- MC\_GETPAR (C macro, mccode-r.h), *134*
- mccode-r (library), **133**
- MCDISPLAY (McStas keyword), 134
- MCNP, 41
- MCSTAS (environment variable), 133, 140
- McStas
  - name, 143
  - pronunciation, 143
- mcstas-r (library), **133**
- mcstas2vitess (McStas tool), 141
- MIN2RAD (constant), **142**
- MNEUTRON (constant), **142**
- monitor\_nd-lib (library), **141**
- Monitors, **100**
  - Beam analyzer, 121
  - Resolution monitor, *see*
  - Samples/Resolution function
- Monte Carlo method, 16
  - accuracy, 20
  - adaptive sampling, 19
  - direction focusing, 19
  - stratified sampling, 19
  
- NAME\_CURRENT\_COMP (C macro, mccode-r.h), *134*
- NORM (C macro, mccode-r.h), *136*
- normal\_vec (C function, mccode-r.c), *136*
  
- Optics, **42, 49, 51**
  - Filter, 45
  - Monochromator, thick, 66
  - phase space transformer, 66
- Output parameter, *143*
  
- PI (constant), **142**
- POS\_A\_COMP (C macro, mccode-r.h), *134*
- POS\_A\_COMP\_INDEX (C macro, mccode-r.h), *135*
- POS\_A\_CURRENT\_COMP (C macro, mccode-r.h), *134*
- POS\_R\_COMP\_INDEX (C macro, mccode-r.h), *135*
- Preprocessor macros, 133
- PROP\_DT (C macro, mcstas-r.h), *134*
- PROP\_GRAV\_DT (C macro, mcstas-r.h), *134*
- PROP\_Z0 (C macro, mcstas-r.h), *133*
  
- RAD2DEG (constant), **142**
- RAD2MIN (constant), **142**
- rand01 (C function, mccode-r.c), *137*
- randnorm (C function, mccode-r.c), *138*
- randpm1 (C function, mccode-r.c), *138*
- randtriangle (C function, mccode-r.c), *138*
- randvec\_target\_circle (C function, mccode-r.c), *138*
- randvec\_target\_rect (C function, mccode-r.c), *138*
- randvec\_target\_rect\_angular (C function, mccode-r.c), *138*
- read\_table-lib (library), **138**
- Removed neutron events, 11, 134
- RESTORE\_NEUTRON (C macro, mcstas-r.h), *135*
- RMS2FWHM (constant), **142**
- ROT\_A\_COMP (C macro, mccode-r.h), *134*
- ROT\_A\_CURRENT\_COMP (C macro, mccode-r.h), *134*
- rot\_apply (C function, mccode-r.c), *136*
- rot\_copy (C function, mccode-r.c), *136*
- rot\_mul (C function, mccode-r.c), *136*
- ROT\_R\_COMP (C macro, mccode-r.h), *134*
- ROT\_R\_CURRENT\_COMP (C macro, mccode-r.h), *134*
- rot\_set\_rotation (C function, mccode-r.c), *136*
- rot\_transpose (C function, mccode-r.c), *136*
- rotate (C macro, mccode-r.h), *136*
- Run-time, *143*
  
- Samples, **67**
  - Resolution function, sample for, 118
- Sampling, 19
- scalar\_prod (C function, mccode-r.c), *136*
- SCATTER (C macro, mcstas-r.h), *134, 135*
- SCATTERED (C macro, mccode-r.h), *135*
- SE2V (constant), **142**
- Setting parameter, *143*
- SHARE (McStas keyword), 133
- Simulation progress bar, 121
- solve\_2nd\_order (C function, mccode-r.c), *136*
- Sources, **22**

- from 1D table input, 45
  - Virtual source from stored neutron events, 117
  - Virtual source, recording neutron events, 117
- `sphere_intersect` (C function, `mccode-r.c`), 137
- Statistics
  - uncertainty, 17
- `STORE_NEUTRON` (C macro, `mcstas-r.h`), 135
- Stratified sampling, 19
- Symbols, 11
- `Table_Free` (C function, `read_table-lib.c`), 139
- `Table_Free_Array` (C function, `read_table-lib.c`), 139
- `Table_Index` (C function, `read_table-lib.c`), 139
- `Table_Info` (C function, `read_table-lib.c`), 139
- `Table_Info_Array` (C function, `read_table-lib.c`), 140
- `Table_Init` (C function, `read_table-lib.c`), 139
- `Table_Read` (C function, `read_table-lib.c`), 139
- `Table_Read_Array` (C function, `read_table-lib.c`), 139
- `Table_Read_Offset` (C function, `read_table-lib.c`), 139
- `Table_Read_Offset_Binary` (C function, `read_table-lib.c`), 139
- `Table_Rebin` (C function, `read_table-lib.c`), 139
- `Table_Value` (C function, `read_table-lib.c`), 139
- `Table_Value2d` (C function, `read_table-lib.c`), 139
- Tools
  - `mcdoc`, 12
- Tripoli, 41
- Units
  - constants and conversions, **141**
- V2K (constant), **142**
- Variance, 17
- Variance reduction, 19
- `vec_prod` (C function, `mccode-r.c`), 136
- Virtual sources, 20
- Vitess, 41
- `vitess-lib` (library), **141**
- VS2E (constant), **142**
- Weight, **17**
  - statistical uncertainty, 17
  - transformation, 18