The MCLib Library: Monte Carlo Simulation of Neutron Scattering Instruments

Philip A. Seeger
239 Loma del Escolar, Los Alamos, NM 87544, USA
E-mail: PASeeger@losalamos.com

Abstract

This report describes the philosophy and structure of MCLib, a Fortran library of Monte Carlo subroutines which has been developed to test designs of neutron scattering instruments. Reports from the proceedings of two meetings [1,2] are incorporated, and later additions are the basis of another proceedings report [3]. A pair of programs (LQDGEOM and MC_RUN) which use the library is shown as an example; a second example using a focusing mirror to increase intensity in a small-angle instrument is also shown. Subroutine abstracts are included in an appendix. The latest release of the source code and documentation should be downloaded from [http://PASeeger](http://PASeeger).

[The former web server at Los Alamos National Laboratory is no longer available.] User input is always requested for additional features to be added to the library.

1. Introduction

Monte Carlo is a method to integrate over a large number of variables. Random numbers are used to select a value for each variable, and the integrand is evaluated. The process is repeated a large number of times and the resulting values are averaged. For a neutron transport problem, we first select a neutron from the source distribution, and project it through the instrument using either deterministic or probabilistic algorithms to describe its interaction whenever it hits something. If it hits a detector, we tally it in a histogram representing where and when it was detected. This is intended to simulate the process of running an actual experiment (but it is much slower). Monte Carlo is a useful supplement to analytical treatment of an instrument, in particular to check and demonstrate “non-intuitive” focusing arrangements, but should never be used as a substitute for thinking. (I am grateful to Jack Carpenter for reminding us of this limitation of Monte Carlo.)

The present MCLib library has been derived from codes written by Mike Johnson at the Rutherford Laboratory [4]. Significant additions and revisions were made by this author in 1984 [5], and the entire code was rewritten in a structured form in 1994 [6]. Whenever the code has been applied to new problems, additions have been made. Several applications of the code have been presented at instrument design workshops [7]. Though not covered in this document, the user interface being developed in collaboration with Thierry Thelliez and Luke Daemen [8] will have a major impact on the usefulness of the package. This allows users to design instruments by defining the locations and properties of a variety of beam elements.


The process is carried out in two stages. First, either the interface or a specific program must be used to generate a description of the geometry of the instrument being simulated. For example,
the program MCLIB may be used to define a small-angle scattering instrument with pinhole collimation, up to three choppers, and an on-axis 2-dimensional position sensitive detector. Essentially all of the user interactions occur in this stage. The output is a geometry file containing the complete problem definition. That file is then passed to a second-stage program, for example MC_RUN which can be downloaded from http://PASeeger.com along with the Monte Carlo library and documentation), which transports neutrons and tallies results in histograms. Principal outputs are a file with a statistical summary, and a data file with histograms of the spectrum and detector. A third stage, which is not part of the Monte Carlo process, is to perform whatever data reduction is appropriate to the experiment being simulated. Some measure of the information content (or a “figure of merit”) is then used to evaluate the design of the instrument. Sample programs that read and plot the output files are provided.

Features of MCLIB that are different from other Monte Carlo libraries include

- Simplified transmission through materials. Rather than compute microscopic interaction in a simple (amorphous unpolarized) region, attenuation of the transmitted neutron is calculated.
- Optics at surfaces. When a neutron reaches a surface, the (complex) index of refraction is computed to decide whether the neutron will reflect or refract.
- Time-dependent devices. There are region types to describe moving devices such as choppers or a gravity focuser.
- Acceleration of gravity is included in transport.
- Polarization is included. This allows algorithms that are different for the two spin states, and precession is computed during transport in a magnetic induction region.
- Scattering functions. Each kind of scattering sample is a region type. The scattering algorithm may be deterministic (reflectometry), probabilistic (hard-sphere scatterer), or a combination (Bragg reflection into a Debye-Scherrer cone).

2. Geometry Description by Surfaces and Regions

The geometry of a system is described by surfaces and regions. A surface is defined by a general 3-dimensional quadratic equation of the form

\[ A x^2 + B x + C y^2 + D y + E z^2 + F z + G + P y z + Q z y + R z x = 0 \]  

(1)

with 10 coefficients, plus a roughness (figure-error) parameter (BETA). The surface divides 3-dimensional space into two parts, which are called the + and − sides of the surface depending on whether the left-hand side of eq. (1) evaluates to a positive or a negative value. For example, a plane perpendicular to the z-axis at z = 1 can be expressed by the equation

\[ z - 1 = 0 , \]

i.e., \( F = 1 \) and \( G = -1 \) (all other coefficients zero). Then all points with \( z < 1 \) are on the − side and all points with \( z > 1 \) are on the + side of the surface. Higher-order surfaces such as toroids, which can not be described by eq. (1), must instead be defined as parameters of a special region (e.g., toroidal mirror, type 14). The scaling of eq. (1) is arbitrary, but we tend to evaluate non-quadratic surfaces as m (coefficients \( B, D, \) and \( F \) dimensionless and \( G \) in m) and quadratic surfaces as \( m^2 \) (coefficients \( A, C, E, P, Q, \) and \( R \) dimensionless, coefficients \( B, D, \) and \( F \) in m, and \( G \) in \( m^2 \)). The parameter BETA is the length of a randomly orientated 3-dimensional vector that is added to the unit vector normal to the mathematical surface to determine the surface
orientation when a particle interacts. For a perfect smooth surface, $BETA = 0$; for $0 < BETA < 1$, $BETA$ is the sine of the maximum angular deviation of the surface normal from smooth. If $BETA < 0$ (or $BETA >> 1$), the surface is completely random.

Note that the coordinate system being used is left handed: the instrument axis is the positive z-direction, the x-axis is horizontal and positive to the right, and the y-axis is vertical with the acceleration of gravity in the negative y-direction.

The geometric shape of each *region* is defined by its relationship to all of the defined surfaces. A positive or negative integer is placed in the region definition if every point in the region is on the + or - side of the corresponding surface, and surfaces which do not bound the region are set to zero. Special characteristics of the boundary are given by the value of the integer: ±1 for an ordinary surface with roughness $BETA$ and the possibility of refraction or critical reflection; ±2 for total reflection; ±3 for diffuse scattering (independent of $BETA$); ±4 for total absorption; ±5 for cases requiring special action (such as a coordinate transformation) whenever a particle enters or leaves the region; and ±6 if the neutron history is to be split after crossing the surface. Generally, no surface may be used as a boundary of a region if any part of that surface is inside the region, because then some points in the region would be on the + side and some on the − side of that surface. Concave (reentrant) shapes are allowed when using quadratic surfaces as in fig. 1a, but the shape in fig. 1b requires that two regions be defined. Because definitions of this sort become very complex when backscattering angles and multiple detectors are used, a special form is provided for embedding one region in another. For example, the enclosing region ("scattering chamber") may be a large sphere. A detector embedded in the scattering chamber is defined in the usual manner, but every one of its surfaces is also made a surface of the scattering chamber with opposite sign and with 10 added to the surface type. (Exceptions are if the surface is already an actual boundary of the scattering chamber, or has already been used with the

Figure 1. Concave regions.

Equations of the numbered surfaces are

1: $x = 0$
2: $x - 4 = 0$
3: $y = 0$
4: $y - 3 = 0$
5: $x^2 - 6x + y^2 - 6y + 14 = 0$ or $(x - 3)^2 + (y - 3)^2 = r^2 = 4$
6: $x = -1.5$
7: $y = -1$

The shaded area in a) can be expressed unequivocally as a single region:

+1  −1  +1  −1  +1  0  0  0

To avoid ambiguity, the area in b) requires two regions, divided as shown by surface 7 (or alternatively by surface 6).

+1  −1  +1  0  0  0  −1
+1  0  0  −1  0  −1  +1

The unshaded area in either case requires 5 regions for a complete unambiguous definition. (One choice of region boundaries is shown by the faint lines.)
opposite sign as now required. In the latter case it is necessary to define separate copies of the
surface for the two embedded regions that use it.) The presence of the 10s digit tells the code to
consider that surface as a possible exit from the enclosing region, but a particle will remain
within the enclosing region if no valid embedded region is found on the other side.

In general any portion of space in which a neutron may move must be defined as being in a
region. The regions should thus be contiguous and non-overlapping. An exception is magnetic
induction regions, which always overlap “real” regions, and do not need to fill all space.

3. Region Types

An element of the instrument consists of one or more regions. Every region has a NAME and a
pointer (INDEX). For a void drift region, INDEX = 0, and for every region which is not a void
INDEX points to a location in a REAL*8 parameter block which contains the region type number,
possibly followed by additional parameters. Future development of the library should be
accomplished by defining new region types and implementing the corresponding algorithms for
how a neutron interacts in such regions (see section 9 below). Defined type numbers are listed
here, and the definitions of parameters are given in Appendix A.

Simple material types:
type 0 = total absorber
type 1 = amorphous unpolarized material (1.1 is polarized)
type 2 = polycrystalline, including Bragg edges: Al, Be diamond, Fe
type 3 = hydrogenous, using S(α,β) tables from MCNP
type 4 = supermirror represented by trapezoidal reflectivity (4.1 polarized)
type 6 = single-crystal filter, Freund formalism

Complex regions:
type 10 = multi-aperture collimator (not yet implemented)
type 11 = multi-slit collimator, vertical blades
type 12 = multi-slit collimator, horizontal blades
type 13 = mosaic crystal, reflection or transmission
type 14 = toroidal mirror
type 15 = multiple identical segments with beam-axis rotation (e.g., curved guides)
type 16 = spin flipper

Time-dependent regions:
type 20.n = chopper (disk or blade)
  20.0 or 20.2 for motion in x-direction, 20.1 or 20.3 for y-direction
  20.2 or 20.3 is counter-rotating (fully closed when edges at 0)
type 21.n = Fermi chopper rotating about the vertical (Y) axis
  21.0, body is transparent and blades opaque to neutrons
  21.1, make estimates of transmission probability
type 22 = gravity focuser
type 23 = removable beamstop

Scattering samples:
type 30.n = small-angle scattering sample
30.0 scatters at constant Q. 30.2 is hard spheres of fixed radius R
30.1 or 30.3 are constant Q or fixed R respectively, with limits on azimuthal angle

type 32.n =

isotropic scatterer with spectrum of fixed energy changes
32.0 scatters with single energy into $4\pi$, 32.1 has limited solid angle,
32.2 has spectrum of energies, 32.3 has both spectrum and limited solid angle

type 34 =
inelastic scattering kernel; no parameters; NAME is ‘[path]filename’ of $S(\alpha,\beta)$ file
in MCNP Type I format (34.1 with limits on azimuthal angle)

type 35 =
scattering from layered reflectometry sample

type 36 =
scattering from isotropic polycrystalline powder (36.1 with limits on azimuthal angle)

Detectors (zero- and one-dimensional):
type 40 =
detector (40.4 is polarized)


type 41 =
vertical linear detector (41.4 is polarized)

type 42 =
horizontal linear detector (42.4 is polarized)


type 44 =
longitudinal linear detector (44.4 is polarized)

Detectors (two-dimensional):
type 43.nm =
2-D detector in various coordinate systems; +0.40 is polarized
43.00, 43.10, 43.20, rectilinear coordinates, respectively (X, Y), (Z, Y), (X, Z)
43.01, 43.11, 43.21, plane polar coordinates ($\rho$, $\phi$), respectively about axes parallel
to Z-, X-, and Y-axes
43.02, 43.12, 43.22, cylindrical coordinates, respectively (Z, $\phi$), (X, $\phi$), (Y, $\phi$)
43.03, 43.13, 43.23, spherical coordinates ($\theta$, $\phi$), with polar axis parallel to Z-, X-, or Y-axes respectively

Scattering chamber:
type 50 =
scattering chamber, void-filled. Other regions may be embedded, indicated by surface
types with 10s digit on. Beam axis may be rotated.

type 51.n =
wrapper for polarization dependence, 2 following regions for spins parallel and
antiparallel to magnetic field
51.0, select spin based on P
51.1, split and track both spins

Magnetic induction regions:
type 60 =
uniform B vector

type 61.n =
gradient of B between 2 surfaces; .1 = linear, .2 = helical interpolation

type 62 =
B vector computed from any number of current loops and point dipoles

type 63 =
solenoidal B

type 64 =
finite current sheets (not yet implemented)

type 65 =
file with vector B to be interpolated (not yet implemented)

type 66 =
multiple connected wire segments

User-defined functions:
types 70-74
available for any function within a region (no geometry)

types 75-79
available for any function, with instrument geometry in the calling sequence

Sources:
type 90.n = source size and phase space to be sampled (*Note: required in every simulation!*)
.1 = first aperture rectangular instead of circular, .2 = second aperture rectangular,
.4 = either/both aperture(s) offset vertically (*Note: options are additive*)

type 91 = fixed (or triangular or square distribution) velocity, or sum of Maxwellians, or “old
style” source energy distribution table and emission time parameters (91.1 is polarized).

type 92 = cubic spline functions for energy and time distributions

type 95 = source file, direct-access binary; no parameters, but NAME must be '[path]filename'
for the file. The file must be opened with logical unit number 95.

4. Program Structures

The relationships of the structures are shown schematically in fig. 2. The library source code is
available in Fortran 90 (F90). Surfaces are defined as a RECORD of type STRUCTURE /SURFACE/,
and element references are of the form SURFACE.G (note the use of “.” instead of “%”
as the member delimiter). Similarly, a geometric region is a RECORD of type STRUCTURE /
REGION/; the structure in a vector IGEOM of 2-byte integers, of length equal to the maximum
allowed number of surfaces. An additional structure, MC_GEOM, contains the numbers of surfaces
and regions in the problem, NSURF and NREG, and arrays of surface and region structures.
Information about the contents of regions is contained in a structure called MC_ELEMENT, which
includes NAME and INDEX arrays, the parameter block PARAM, and the pointer NEXTINDEX to
the next available location in PARAM. The surface coefficients and the PARAM block are
REAL*8.

The final structure is PARTICLE, which includes the position, velocity, time of flight, mass (1 for

![Figure 2. Structures used in MCLIB.](image)
a neutron), charge (0 for a neutron), statistical weight, and beam polarization vector. All values are REAL*4. A purely “analog” Monte Carlo traces each individual neutron until it is either lost or detected. MCLIB uses “weighted” neutrons, and in many of the processes the statistical weight is multiplied by the probability of survival instead of using a random number to decide whether to terminate the history (“Russian Roulette”). This is especially beneficial when scattering probability is small, as in subcritical reflection. To track more long-wavelength neutrons (which in general have larger scattering probability), the source distribution used is $\lambda^2 I(\lambda)$ instead of $I(\lambda)$ and the initial weight is proportional to $1/\lambda^2$. The tallied results are then the sum of detected neutron weights. The relative error in each bin, however, depends on the number of histories recorded.

Neutron beam polarization is represented by a vector $\mathbf{P} = (P_X, P_Y, P_Z)$ with magnitude $P \leq 1$. The probability that any particular neutron in the beam will interact with spin parallel to a unit vector $\mathbf{n}$ is $(1 + \mathbf{n}\cdot\mathbf{P})/2$ and the anti-parallel probability is $(1 – \mathbf{n}\cdot\mathbf{P})/2$. In the simulation, a component may either split the neutron into two independent histories with statistical weights multiplied by the respective probabilities, or it may use a random number to select one spin or the other based on the relative probabilities. Any operation that is symmetric with respect to spin, such as precession or spin-flip, can be applied either to the combined or to the separated histories. For algorithms that act differently on the two spin states, the single history must be decomposed; after the interaction, the two histories may either remain separated or (only if the directions of $\mathbf{P}$ are the same) may be recombined.

The structure definitions are found in the file MC_GEOM.INC in Appendix A. The dimensions of the arrays (since 14 Jun 2005) are $MAXS = 400$, $MAXR = 255$, and $MAXP = 8000$. An additional parameter is the allowed number of magnetic regions, $MAXM = 50$.

5. Subroutine Library

Complete descriptions, calling sequences, revision history, and externals of all of the library subroutines are given in Appendix B; the most up-to-date listing of the subroutine abstracts and of the source codes are at http://PASeeger.com. The following listing divides the subroutines into (somewhat arbitrary) categories, and includes the latest modification date and an abbreviated description. Note that if the operating system does not include the function RAN(ISeed), then such a REAL*4 function must be provided as part of the library.

Vector analysis and transport

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANGLI</td>
<td>14 Sep 98</td>
<td>angle of incidence</td>
</tr>
<tr>
<td>ANGTORUS</td>
<td>02 Aug 03</td>
<td>angle of incidence on torus</td>
</tr>
<tr>
<td>CAPILLARY</td>
<td>18 Sep 03</td>
<td>angle of incidence on torus</td>
</tr>
<tr>
<td>DIST</td>
<td>11 Sep 06</td>
<td>distance to a surface</td>
</tr>
<tr>
<td>DIST_A</td>
<td>20 Sep 03</td>
<td>distance to surface, including acceleration</td>
</tr>
<tr>
<td>DISTORUS</td>
<td>14 Sep 98</td>
<td>distance to a toroidal shell</td>
</tr>
<tr>
<td>DTOEX</td>
<td>18 Aug 06</td>
<td>distance to nearest boundary</td>
</tr>
<tr>
<td>ELSCAT</td>
<td>28 Sep 98</td>
<td>do elastic scattering</td>
</tr>
<tr>
<td>ELSCAT2</td>
<td>28 Sep 98</td>
<td>elastic scattering, biased azimuth</td>
</tr>
<tr>
<td>LIGHTRFL</td>
<td>14 Sep 98</td>
<td>light reflection probability</td>
</tr>
<tr>
<td>LMONOCR</td>
<td>16 May 06</td>
<td>reflection by crystal monochromator</td>
</tr>
<tr>
<td>LREFLCT</td>
<td>08 Mar 99</td>
<td>neutron reflection probability</td>
</tr>
<tr>
<td>MOVEA</td>
<td>20 Sep 03</td>
<td>move with acceleration</td>
</tr>
<tr>
<td>MOVEX</td>
<td>03 Apr 04</td>
<td>move a particle</td>
</tr>
<tr>
<td>NEXTRG</td>
<td>04 May 02</td>
<td>find region across boundary</td>
</tr>
</tbody>
</table>
interactions at surfaces

vector orthogonal to given 3-vector

get source neutron

get phase space and brightness of source

find what happens to particle within region

find what happens to particle leaving region

photon at region boundary

do reflection

apply Snell's law

find if within region

dummy routines for user-defined functions

cross product of two 3-vectors

find what region particle is in

angle of incidence at wavy surface

attenuation of Al (default 300 K)

attenuation of Be (default 100 K)

attenuation of Fe (default 300 K)

attenuation of single-crystal filter

attenuation of diamond/Co (default 300 K)

impulse approximation

inelastic scattering kernel

probability from spherical scatterer

scatter from polycrystalline powder

reflection probability from multiple layers

resonance scattering length and cross sections

single crystal, find reciprocal lattice

scatter from crystal

Fermi chopper

gavity focuser

disk or blade chopper

magnetic vector potential of a current loop

magnetic vector potential for solenoid

find B-vector

set the Runge-Kutta precision

family of field algorithms

Romberg integration of |B|

B field from single current loop

find magnetic region number

find magnetic region number

map B vector along trajectory

move with spin precession

projection of polarization on magnetic field

stepsize manager for Runge-Kutta

4th-order Runge-Kutta for Bloch eqn.

induction field due to solenoid.

random unit vector in limited solid angle

random unit vector in 4\pi

probability from convolution of top-hats
Subroutine OPERATE and its other entry point EXIT_REG play special roles in the transport process. As can be inferred from the abstract in Appendix B, this subroutine is called when a neutron enters a new region to determine what happens. Possible results on exit from OPERATE include detection or loss of the neutron, exiting the region with reduced statistical weight (partial absorption), splitting the neutron into a transmitted and a scattered particle with the sum of statistical weights equal to the original weight, or a transform of the coordinate system of the problem. Coordinate transforms are applied for region types made up of sub-regions (e.g., Soller slits and benders, and multi-segment guides), elements that change the beam direction (benders and monochromators), and time-dependent devices (gravity focuser). In order to assure that the coordinate system is properly restored when the neutron leaves the region, EXIT_REG must be called; this is flagged by using ±5 as the integer defining all exit surfaces from such a region.
6. Analysis Routines

In general, the user will be expected to use his/her own analysis routines, the same as if the data originated from a real instrument. This will generally require a conversion of the data formats. Sample programs to convert .DAT output files are available from the author [please contact by e-mail at PASeege@losalamos.com]. Specifically, the program SWITCH will read a 1D file and convert it to column format, and the program SPREAD will convert 2D data to a spreadsheet format. For 3D files, the sample program QLfromXYt is given; this is a specific application for small-angle scattering, based on the data reduction program SMR that is used for the LQD instrument at Los Alamos. Any of these may be modified to fit a user’s needs. For visualization of the data, programs QUIKPLOT and QUIK2D are provided for 1D and 2D files respectively, using the PGPLOT graphics library, free from ftp://astro.caltech.edu/pub/pgplot/.

Some additional subroutines in the analysis folder that are not included in MCLIB itself are

- DLfromZt 30 Oct 09 reduce powder diffraction data (elastic)
- EEfromRt 30 Dec 08 reduce inverse-geometry inelastic data
- GAUSSFIT 18 Feb 85 least-squares fit Gaussian to data
- LUBKSB 30 Aug 89 linear equation solution by lower\upper backsubstitution
- LUDCMP 21 Apr 93 lower\upper decomposition
- NEWHEAD 25 Mar 98 concatenate strings before and after operation string
- NEWOP 21 Jan 88 attach operation character to operation string
- OPLIST 18 Mar 00 concatenate 2 operation strings with operator
- PAINT 04 Jul 98 output a 2D plot with PGPLOT drivers
- PLOT10 06 Oct 92 convert most (38) PLOT10 primitive calls to PGPLOT
- CHECK_TEXT decode PLOT10 (DISSPLAY) special characters to PGPLOT
- CONTOUR contour plot of an array
- CURVE plot with symbols and various dashed lines
- ERRBARS plot linear or log error bars
- LINAXIS scale ends of linear axis to rounded values
- LOGAXIS scale ends of logarithmic axis to rounded values
- POLYFIT 27 Jan 93 least-squares fit a polynomial to data
- PREMUL 18 Mar 00 attach multiplier to front of operation string
- QBINS 04 Sep 89 generate Q-bin boundaries on log/lin/log scale
- QLfromXYt 18 Oct 00 reduce small-angle data
- RBAR 15 Jun 94 average radius of a rectangular pixel
- SPLINE 24 Jun 96 fit cubic spline to points
- SPLINT 24 Nov 84 interpolate using cubic spline
- SPREAD 23 Sep 97 convert 2D Block ASCII data to spreadsheet format
- SWITCH 19 Apr 08 convert 1D Block ASCII data to column format
- WRTABSTR 18 Oct 00 write an LQD standard file abstract

7. Creating a New Type

The process of adding toroidal mirrors will be used as a case study. There were four steps:
1. Define a new region type (e.g., 14) and assign parameters and names in MC_ELMNT.INC. (Note that types 70-79 are reserved for user development.)
2. Develop needed algorithms and procedures (DISTORUS and ANGTORUS).
3. Insert the “methods” for the new type in subroutine OPERATE.
4. Test, debug, and refine the previous three steps.

1. There is renewed interest in the use of focusing mirrors in neutron instruments, for example the proposal of Benno Schoenborn for a Laue protein-crystallography instrument [10], and a recent paper by John Copley [11]. Although an ellipsoid would be the optically preferred
shape, a toroid may be more practical to manufacture. Since a toroid can not be defined as a surface by eq. (1), it was necessary to define it as a region. The nature of the region is that it is divided into two subregions by the toroidal surface. The ten parameters can be seen in Appendix A at type 14. The torus is defined by its major radius $R$, the radius $r$ of the generating circle, the offset of the center of the generating circle from the beam axis, and the longitudinal position of the center of revolution. Also, the sign of the major radius shows which side of the beam axis the center is on, and the sign of the minor radius is used as a flag for orientation: $+$ for horizontal and $-$ for vertical. (The choice of parameter three as a small offset instead of distance from the center of the torus was crucial to maintaining precision.) The fifth parameter is surface roughness, and then three parameters are provided to define a rotation of the instrument beam axis after reflection. Finally, there are two pointers to define the materials inside and outside the torus. The “inside” material is in front of the mirror surface, and is usually void, indicated by a pointer of zero. The “outside” is the mirror surface/substrate, and the pointer is the offset of its definition in the PARAM block relative to the torus parameters. E.g., if the material type immediately follows this block, then the pointer value is 12 (or 1 + NUMBER_14, the number of cells in PARAM used by type 14).

2. Finding the intersections of a line with a torus (a quartic surface) is not a trivial problem. Previous codes [14,15] have solved the problem very generally with sophisticated methods to retain precision. My approach was to begin with three limiting assumptions and one complication:
- only the near half of the toroid is relevant
- only the outer (concave) half-shell of the toroid is relevant
- only real solutions for the intersections are relevant
- the particle trajectory is a gravitational parabola instead of a straight line.

At this stage I had the assistance of Mike Fitzsimmons and Tom Klugel (MLNSC) to study methods of setting up and solving the fourth-order equation. The first method we used to set up the equation was copied from [12], using analytic derivatives of the trajectory in the coordinate system of the torus. This ran into serious difficulties when gravity was included (see §4 below), and the eventual solution was first to bracket the nearest and furthest intersections (if any!) of the parabola with bounding cylinders and planes of the relevant portion of the torus, and to solve for the distance from the surface parametrically vs. time of flight at five equally spaced points. Since the spacing of the points is uniform, multiplication by a predetermined constant matrix solves for the coefficients of the fourth-order equation of nearest distance as a function of time. One or two real roots (if any) are then found by the “safe” Newton-Raphson method discussed in Numerical Recipes [13], which uses bisection to keep the roots bracketed whenever the iteration step would move outside the current bracket. For further details, see the source code for DISTORUS.F.

Finding the angle of incidence, subroutine ANGTORUS, is far simpler. It is essentially a copy of WOBBLE with derivatives of the torus equation instead of eq. (1). As in DISTORUS, precision is enhanced by accounting for the fact that the major radius of the torus is large compared to any other distance.

3. Inclusion of type-specific code in subroutine OPERATE requires adherence to the overall structure of the routine, and will probably be the most difficult (or dangerous) aspect for a user to add a function to MCLIB. Here is the scenario:
• When a particle enters a region, `OPERATE` is called with the following references:

  - `PART` = record containing description of particle (input/output)
  - `EXDIST` = distance to exit surface particle is aimed at (m) (input/output)
  - `PARAMS` = array with description of what is in this region (input)
  - `GEOM` = structure with all surface and region geometry definitions (input)
  - `ELEMENT` = structure with all region space and magnetic parameters (input)
  - `IREG` = region number of device, or subregion within device (input/output)
  - `JSURF` = surface number, if particle is initially on surface (input)
  - `KSURF` = surface number that particle is pointed toward (input/output)
  - `NAME` = name of region, may used as file name (e.g., type 34) (input)
  - `TRANSMIT` = flag to compute transmission of sample types 30-39 (input)
  - `TRACE` = ‘P’ or ‘E’ to trace intermediate trajectories on Plan or Elevation view
  - `FIELDS` = flag to include magnetic fields (input)
  - `FLAG` = flag set to .FALSE. if (e.g.) chopper in wrong frame (output)
  - `PART_2` = description of particle created by operation (output)
  - `DET_WT` = statistical weight of detected particle (output)
  - `IX, IY` = position bin numbers of detected particle (output)
  - `ISEED` = random-number generator seed (input/output)

• `OPERATE` determines the region type from `PARAMS(1)` and transfers to the appropriate block of code in an `IF-THEN-ELSE IF` structure (this will become a `CASE` structure when F90 is more widespread).

• The code may use any variables marked “input” and may change any variables marked “output” in the above list. Local variables will have to be declared, and any variables to be retained across entries must have distinctive names and be placed in a `SAVE` statement.

• Parameter names from `MC_ELMNT.INC` and physical constants from `CONSTANT.INC` (see Appendix A) are available for use.

• Valid operations include: moving the particle to the exit from the region with possible reeducation of statistical weight to account for absorption (set `EXDIST` to 0); selection of a subregion without moving; reflection (set `KSURF` negative); rotate the coordinate system; statistically split the particle or create a new one (in `PART_2`); scatter either elastically or inelastically; multiple scattering, keeping control through several motions of the particle till it reaches the region exit; detect the particle and find encoded position in detector. Examples of all these possibilities can be found in `OPERATE.FOR`.

• Algorithms may be deterministic or probabilistic (or both). Various random deviate functions are available (and others may be added to the library). For single random numbers, use `RAN(ISEED)`, or `RAN0(ISEED)` to eliminate correlations.

• For compatibility with older Fortran compilers without recursive subroutines, the code should not make any calls to `OPERATE` for processing subregions. Use no numbered statements. Style should be similar to the existing `OPERATE.FOR` code.

In the case of the toroidal mirror, the actions are to determine the distance to the torus and which side the particle is on using `DISTORUS`, and to move either to that surface or the region exit (whichever is closer) through the appropriate material, compute reflection probability using `ANGTORUS` and the ratio of scattering length densities across the surface, and repeat all of these actions till the region is exited. Note that only material types 0 (void), 1 (amorphous non-magnetic), or 4 (supermirror layer) are allowed, due to the restriction that all code for the type must be inline. In fact, the code for type 1 and type 4 regions has been copied into this block.
4. The debugging of this module is an example of cooperation with outside users. The code had been tested by reproducing the results in [11], but in a different coordinate system. When John Copley tried to use the code, there were subtle errors. It took one more test to show that my code was at fault, rather than John’s interpretation of it, two more rounds of modification and testing to uncover a serious philosophical flaw in the algorithm, and two more rounds to “perfect” the code. This version passes all tests, and is twenty times faster [14] than the program used previously.

8. A Focusing Mirror in a Small-Angle Scattering Instrument

The original paper on curved-mirror neutron optics [15] suggested small-angle scattering as an application. The authors considered a bent mirror forming a cylinder with axis transverse to the beam, and a narrow slit aperture with the solid angle of the mirror matched to the solid angle of the neutron source. The same geometrical matching can be applied to illuminate a mirror that focuses in two dimensions, as illustrated in Fig. 3. (Note that the transverse dimensions in Fig. 3 are exaggerated by a factor of 10 relative to the longitudinal dimensions.) The MCLIB code allows us to compare mirror shapes in this geometry: the ideal ellipsoid, the tangent toroid, cylindrical segments (with longitudinal cylinder axes), or toroidal segments (bent cylinders with varying curvatures). Compared to a two-aperture collimation system with the same resolution, a factor of 10 increase in intensity is possible because the full illuminated surface of the moderator may be viewed. The tradeoff between intensity and resolution depends on the single aperture.

The mirror considered as an example was 3.0 m long and 75 mm wide, with a major axis of 9 m and transverse (minor) axes of 157 mm. The glancing is 1º at the center (beam axis bent by 2º). The radius-of-curvature parameters of the tangent toroid [11] are \( R = 257.77 \) m and \( r = 78.54 \) mm. The source aperture had a diameter of 2.0 mm, located at one focus of the ellipsoid; at this eccentricity, the focus is less than 1 mm from the vertex. Also, the transverse diameter of the ellipsoid at the focus is only 2.7 mm, so the source is “large” in terms of optical properties, and Monte Carlo is a useful tool for estimating resolution at the detector placed at the other focus. The detector pixel size was made small (0.25 mm) and the encoding uncertainty was zero so as not to affect the computed resolution when compared to the source aperture size, which contributes 0.50 mm to the standard deviation in each detector coordinate. Since the effect of gravity is always included in the algorithms of MCLIB (for any particles with rest mass), it is necessary to place the source aperture (and the detector center) below the instrument axis if the reflection

![Figure 3. Small-Angle Scattering instrument using a focusing mirror for increased intensity.](image-url)
plane is horizontal by a distance which is proportional to wavelength, such that the nominal trajectory at the mirror center is horizontal. However, if the reflection is upwards (mirror horizontal and concave up), then no first-order correction is necessary because the excess downward velocity acquired before striking the mirror is reflected and then canceled by downward acceleration after reflection. (This cancellation is exact only for neutrons reflected at the center of the mirror, but is independent of wavelength.) For a white source (i.e., spallation) the vertical reflection plane is much to be preferred.

A comparison of the beam spots for the ellipsoid and toroid is shown in Fig.4, for vertical reflection of a cold-moderator spectrum from 2–20 Å. (A supermirror reflecting layer was assumed, and reflectivity is good above 3 Å.) The standard deviations for the ellipsoid are 0.64 mm in both the horizontal and vertical directions, and those for the toroid are 1.05 mm horizontal and 2.29 mm vertical. A minimum Q-value less than 0.001 Å\(^{-1}\) could be achieved with this geometry in an instrument of total length 13 m (with the ellipsoidal mirror). Further details of this simulation have been presented elsewhere [16].

![Figure 4](image)

Figure 4. Images of a 2-mm diameter aperture for ellipsoidal (left side) and toroidal (right side) focusing mirrors. The reflection plane is vertical (upward) to correct for gravity. Each pattern is a histogram for 250000 detected neutrons with wavelength distribution from 2–20 Å, from a coupled liquid-hydrogen moderator. Each figure is 20 mm square on the detector (printed at three times size), and the intensity scale is logarithmic from 1 to 93250 counts per pixel.

12. Future Directions

It is hoped that the MCLIB code will be generally useful to designers of neutron instruments. For this goal to be realized, two sets of improvements are being pursued. First, a general user interface was created on a web page at Los Alamos National Laboratory [since dropped, now available for Windows users at http://PASeeger.com] so that geometry files may be constructed more easily than described in this document. The overall project is called the Neutron Instrument Simulation Package (NISP). Features of MCLIB are continually being implemented in the web page, and the MCLIB library is being modified as necessary to support the NISP project. Second, we are soliciting input of ideas, algorithms, and/or (public) code modules from interested or potential users, so that NISP will do what you want. All code included will be acknowledged in the subroutine listings and abstracts. Please communicate with the author by e-mail to PASeeger@losalamos.com if you have any questions or suggestions.
Acknowledgments

This effort has been supported historically by the Los Alamos National Laboratory, and the author is grateful to Rex Hjelm, Luke Daemen, and the Manuel Lujan Jr. Neutron Scattering Center for their continued interest and financial support. The Manuel Lujan Jr., Neutron Scattering Center is a national user facility funded by the United States Department of Energy, Office of Basic Energy Sciences-Materials Science, under contract number W-7405-ENG-36 with the University of California. It is the intention of the author that MCLIB remain in the public domain. Up to 2000, the copyright was assigned to the Regents of the University of California (see notice in Appendix A). The most recent versions of this document and the library and auxiliary codes as described above will continue to be available to any interested users from the author. [Latest version at http://PASeeger.com, no longer kept at the Manuel Lujan Jr. Neutron Scattering Center.] The code remains completely open source, and we follow in spirit the rules of the gnu General Public License [http://www.gnu.org/licenses/gpl.txt], which is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We require that any such use of the routines carry full author attribution as found within the source codes.

References


Appendix A

Files MC_ELMNT.INC, MC_GEOM.INC, MCGEOM90.INC, and CONSTANT.INC

C******* INCLUDE FILE, 'MC_ELMNT.INC' **********
C
C Copyright, 2000, The Regents of the University of California.
C This software was produced under a U.S. Government contract (W-7405-
C ENG-36) by the Los Alamos National Laboratory, which is operated
C by the University of California for the U.S. Department of Energy.
C The U.S. Government is licensed to use, reproduce, and distribute
C this software. Permission is granted to the public to copy and use
C this software without charge, provided that this notice and any
C statement of authorship are reproduced on all copies. Neither the
C Government nor the University makes any warranty, express or implied,
C or assumes any liability or responsibility for the use of this soft-
C ware.
C
C Philip A. Seeger, under rules of the GNU General Public License,
C http://www.gnu.org/licenses/gpl.txt, which is hereby incorporated
C in this document.
C
C Definitions of beam elements which may occur in regions, and their parameters
C
C P. A. Seeger, April 20, 1994
C 04 Jan 1995: define offsets of parameters within blocks, rather than
C structure with UNIONs [PAS]
C 10 Jan 1995: modified source type 90; added type 91 for source spectrum
C and lineshape description [PAS]
C 01 Feb 1995: added types 5 (Be), 13 (crystal monochromator), and 35
C (reflectometry) [PAS]
C 15 Feb 1995: modified type 91; moved structure definition to MC_GEOM.INC
C [PAS]
C 04 Mar 1995: types 90.n for rectangular and/or offset phase space [PAS]
C 09 Mar 1995: type 44 (longitudinal detector); all detectors need surface
C number [PAS]
C 06 Jun 1995: 2 more parameters in monochromator type 13 [PAS]
C 04 Aug 1995: type 6, single-crystal filter [PAS]
C 26 Aug 1995: type 32, isotropic scatterer [PAS]
C 05 Sep 1995: revised parameters for pulse shape (type 91) [PAS]
C 16 Oct 1995: changed parametrization of supermirror (type 4) [PAS]
C 11 Nov 1995: add NUMBER_nn to all types [PAS]
C 22 Nov 1995: add subtypes and coordinate origin to type 43 [PAS]
C 24 Jan 1996: type 95, source from direct-access binary file [PAS]
C 05 Aug 1996: type 14, toroidal mirror [PAS]
C 01 Jul 1997: revised Ikeda-Carpenter form for type 91 [PAS]
C 03 Jul 1997: changed subtypes of type 43 (2 digits) [LLD,PAS]
C 04 Nov 1997: replace type 31 with subtype 30.1 [PAS]
C 02 Dec 1997: subtype 32.1 with limited solid angle [LLD]
C 08 May 1998: subtype 32.2 with multiple energies [LLD,PAS]
C 12 May 1998: *WARNING* types 30 & 36 parameter order changed; subtype 30.1
C changed to 30.2; new subtypes 30.1, 34.1, and 36.1 [PAS,LLD]
C 26 May 1998: type 35, changed ROUGHNESS to RUF_LAYER [PAS]
C 26 Jul 1998: added type 21 (Fermi chopper) [PAS,LLD]
C 15 Sep 1998: *WARNING* change eV to meV, types 32 and 90 [PAS]
C 07 Jan 1999: type 15, multiple segments with rotation [PAS]
The first entry in PARAM for each element identifies the type, followed by a varying number of parameters.

INTEGER ELMNT_TYPE
PARAMETER (ELMNT_TYPE=0)
C type 0 = total absorber; no additional parameters
INTEGER NUMBER_00
PARAMETER (NUMBER_00=1)
C type 1.0 = amorphous unpolarized material; 4 parameters
C .1 = amorphous magnetic material; 5 parameters
C .2 = amorphous unpolarized, with C.M. scattering correction
C 1,2) Real and Imaginary scattering-length density (10^10/cm^2)
C 3) macroscopic scattering cross section (1/m)
C 4) velocity-dependent macro cross section, at 1 m/us (1/us)
C 5) (for 1.1 only) magnetic scattering-length density (10^10/cm^2)
C 5) (for 1.2 only) atomic mass/Mn
INTEGER REAL_RHO, IMAG_RHO, NSIGMA0, NSIGMAV, RHO_MAG, SCAT_AW, &
& NUMBER_01
PARAMETER (REAL_RHO=1,IMAG_RHO=2,NSIGMA0=3,NSIGMAV=4,RHO_MAG=5, &
& SCAT_AW=5,NUMBER_01=6)
C type 2.n = polycrystalline w/Bragg edges; 2 additional parameters
C .0 defaults to Al, 300K, no parameters
C .1 Al, 300K
C .2 sintered diamond/Co, 300K
C .3 Fe, 300K
C .5 Be, 100K (NOTE: same as material 5.1)
C 1) temperature (K)
C 2) atomic mass/Mn, for C.M. correction
INTEGER MAT_TEMP, MAT_AW, NUMBER_02
PARAMETER (MAT_TEMP=1,MAT_AW=2,NUMBER_02=3)
C type 3.n = S(alpha,beta) materials from MCNP; 10 parameters
C .1 = Polyethylene, 300K
C .2 = Water, 300K
C  .3 = Be, 300K
C  .4 = Be, 100K
C  .5 = Graphite, 300K
C 1-5) same as type 1.2, for the light atom (H or Be)
C 6-9) same as 1-4) of type 1, for sum of other atoms
C 10) number density of light atom in the mixture (10^22/cm^3)
C    1-5) same as type 1.2, for the light atom (H or Be)
C    6-9) same as 1-4) of type 1, for sum of other atoms
C    10) number density of light atom in the mixture (10^22/cm^3)

INTEGER REAL_RHO2, IMAG_RHO2, NSIGMA02, NSIGMAV2, ATOM_DENS, &
    NUMBER_03
PARAMETER (REAL_RHO2=6,IMAG_RHO2=7,NSIGMA02=8,NSIGMAV2=9, &
    ATOM_DENS=10,NUMBER_03=11)

C  type 4.0 = supermirror represented by trapezoidal reflectivity; 4 parameters
C  .1 = supermirror with magnetic properties; 5 parameters
C  1,2) Real & Imaginary scattering-length density (10^10/cm^2) (see type 1)
C  3) Supermirror multiplier, average of Up & Down spin states
C  4) Reflectivity at maximum supermirror limit
C  5) (for 4.1 only) (Up_multiplier - Down_multiplier)/2
C    INTEGER SUPER_MULT, SUPER_REFL, SUPER_MAG, NUMBER_04
PARAMETER (SUPER_MULT=3,SUPER_REFL=4,SUPER_MAG=5,NUMBER_04=6)

C  type 5.n = beryllium, including Bragg edges; 1 additional parameter
C  NOTE: this material type should be replaced by 2.5
C  1) for type 5.1, temperature (K); default 100K for type 5.0
C    INTEGER NUMBER_05
PARAMETER (NUMBER_05=2)

C  type 6 = single-crystal filter, Freund formalism; 3 parameters
C  xsec = sigfree*(1-exp(-C2/lambda^2)) + sigabs*lambda
C  1) Limiting (short wavelength) free-atom macroscopic cross section (1/cm)
C  2) -ln(1 - (sig(1A)-sigabs)/(sigfree-sigabs)) (A^2)
C  3) sum of 1/v macroscopic cross sections at 1A (1/cm/A)
C    INTEGER XSIGFREE, X_C2, XSIGABS, NUMBER_06
PARAMETER (XSIGFREE=1,X_C2=2,XSIGABS=3,NUMBER_06=4)

C  type 7.n = nuclear resonance(s), Lynn general 1-channel reduced R-matrix
C  formalism; 3+10*n parameters
C  .0, nuclear spin 0, or no polarization
C  .1, two spin states, polarization allowed
C  1) Number of isotope-spin combinations
C  2) Channel radius, a0 (10^-12 cm)
C  3) Pivot energy for 2nd-order expansion of non-resonant terms (eV)
C  4) Isotopic nuclear density * spin-weight factor (atoms/(b-m));
C    for 7.1, >0 for channel spin j+1/2 (ortho) and <0 for j-1/2 (para)
C  5) Resonance energy level, E (eV)
C  6) Neutron width/(2 k a0), with k=0.00219667*sqrt(E) (eV)
C  7) Reaction width (eV)
C  8,11) real and imaginary constant terms, A
C  9,12) real and imaginary linear terms, B (/eV)
C  10,13) real and imaginary quadratic terms, C (/eV^2)
C  14-23) repeat 4-13 for next isotope-spin, etc.
C    INTEGER LYNN_NIS, LYNN_A0, LYNN_EH, LYNN_FG, LYNN_EL, LYNN_GNR, &
        LYNN_GR, LYNN_AR, LYNN_BR, LYNN_CR, LYNN_A1, LYNN_B1, &
        LYNN_CI, NUMBER_07
PARAMETER (LYNN_NIS=1,LYNN_A0=2,LYNN_EH=3,LYNN_FG=4,LYNN_EL=5, &
        LYNN_GNR=6,LYNN_GR=7,LYNN_AR=8,LYNN_BR=9,LYNN_CR=10, &
        LYNN_A1=11,LYNN_B1=12,LYNN_CI=13,NUMBER_07=14)

C  type 10 = multi-aperture collimator (not implemented)
C  type 11.n = multi-slit collimator, vertical blades; 7 parameters
C  .0 = straight, .1 = bender, .2 = radial
C  the type 11 region is followed by up to 5 interior regions
C  1) Spacing of slits, centerline-to-centerline at entrance (m or rad)
C  2,3) X,Z at entrance of the region OR center of radial collimator (m)
C  4) (.0) Rate of divergence (>0) or convergence (<0) of one slit OR
C 4) (.1) Curvature (1/radius) of cylindrical centerline (1/m) OR
C 4) (.2) Azimuthal angle (from +Z) of center line of radial array
C 5) Number of slots in the collimator
C 6,7) (.0) For a straight collimator, sine and cosine of tilt angle from
C Z-axis, measured CCW (positive tilt is to the left) OR
C 6,7) (.1) For a curved system, sine and cosine of half the bend angle
C measured CCW (positive bend is to the left) OR
C 6,7) (.2) For a radial collimator, sine and cosine of slit width
INTEGER C_DELTA, C_XCENTER, C_ZCENTER, C_TAPER, C_NSLOTS, &
  C_SIN_PHI, C_COS_PHI, NUMBER_11
PARAMETER (C_DELTA=1, C_XCENTER=2, C_ZCENTER=3, C_TAPER=4, C_NSLOTS=5, &
  C_SIN_PHI=6, C_COS_PHI=7, NUMBER_11=8)
C  type 12 = multi-slit collimator, horizontal blades; 7 parameters
C  Same parameters as type 11. Slope & bend angles positive upward.
C INTEGER C_YCENTER, NUMBER_12
PARAMETER (C_YCENTER=2, NUMBER_12=8)
C  type 13 = reflecting mosaic crystal, Bragg or Laue mode; 18 parameters
C 1) Twice the crystal plane spacing (\AA)
C 2) (\delta d)/d = rms spread of Gaussian plane spacing if \geq 0, or
C fwhm of Lorentzian if <0
C 3) Horizontal mosaic spread (about vertical axis), rms of sine of angle
C 4) Vertical mosaic spread about horizontal axis in crystal surface
C 5-7) Orientation cosines of reflecting plane normal vector, with crystal
C placed normal to z-axis; sin(chi)cos(psi), sin(chi)sin(psi), cos(chi),
C where chi is angle of TAU from +Z-axis and psi is in the X-Y plane.
C 8) Thin-crystal step length through the crystal, t (m)
C 9) Reflectivity per step*n/(n/lambda^2), t d |F|^2/(V0^2 cos(th)) (/A^2)
C 10-12) Absorption factors = bound macroscopic cross sections x t:
C 10) coherent inelastic, to be multiplied by Einstein form factor;
C 11) incoherent scatter (constant); 12) 1/v absorption at 1/A (/A)
C 13) 4 pi u0, where u0 = rms atom displacement (\AA), for coh. inelas. form
C 14,15) Nominal (X,Z) position for rotation of instrument axis (m)
C 16,17) Sine and cosine of take-off angle (2 theta); sine=0 for no rotation
C 18) Square of distance to move from origin after rotation (m^2)
INTEGER M_2D_SPACE, M_D_SPREAD, HMOSAIC, VMOSAIC, TAU0_X, TAU0_Y, &
  M_THIN, M_REFL, M_INELAS, M_ABSRB, &
  M_4PIU0, M_X0, M_Z0, M_SIN_2TH, M_COS_2TH, M_RSQ, NUMBER_13
PARAMETER (M_2D_SPACE=1, M_D_SPREAD=2, HMOSAIC=3, VMOSAIC=4, &
  TAU0_X=5, TAU0_Y=6, TAU0_Z=7, M_THIN=8, M_REFL=9, &
  M_INELAS=10, M_ABSRB=12, M_4PIU0=13, M_X0=14, &
  M_Z0=15, M_SIN_2TH=16, M_COS_2TH=17, M_RSQ=18, &
  NUMBER_13=19)
C  type 14 = toroidal mirror; 10 parameters
C 1) Radius of rotation of torus, from axis to center of generating
C circle; positive if torus axis right of or above beam axis (m)
C 2) Radius of cross section of torus; negative if torus vertical (m)
C 3) Offset of center of generating circle from beam axis (m)
C 4) Z-coordinate of torus axis (m)
C 5) Surface roughness parameter
C 6-8) sin, cos, and Z-center for beam axis rotation (if any)
C 9,10) Offsets to parameter blocks for interior and exterior regions,
C zero if void; types 1 and 4 are supported
INTEGER TORUS_A, TORUS_B, TORUS_D, TORUS_Z, TOR_BETA, TOR_SIN_TH, &
  TOR_COS_TH, TOR_Z_ROT, INSIDE_OFFSET, OUTSIDE_OFFSET, &
  NUMBER_14
PARAMETER (TORUS_A=1, TORUS_B=2, TORUS_D=3, TORUS_Z=4, &
  TOR_BETA=5, TOR_SIN_TH=6, TOR_COS_TH=7, TOR_Z_ROT=8, &
  INSIDE_OFFSET=9, OUTSIDE_OFFSET=10, NUMBER_14=11)
C  type 15 = multiple segments with rotation; 7 parameters
C 1) Number of segments
MCLIB

2) Chord length across one segment at X=0 (m)

3,4) Sine and cosine of segment wedge angle (positive CCW, left)

5) Z-coordinate of entrance plane (m)

6) L sin(\theta/2) = negative X-offset of origin when rotated (m)

7) L cos(\theta/2) = Z-offset of rotated origin (m)

INTEGER N_SEGMENTS, SEG_LENGTH, SEG_SIN, SEG_COS, SEG_Z0, SEG_DX, &
& SEG_DZ, NUMBER_15

PARAMETER (N_SEGMENTS=1, SEG_LENGTH=2, SEG_SIN=3, SEG_COS=4, SEG_Z0=5, &
& SEG_DX=6, SEG_DZ=7, NUMBER_15=8)

type 16 = spin flipper; 6 parameters

1-3) direction cosines of flipper rotation axis

4-5) sine and cosine of spin rotation, right-handed about axis

6) nominal neutron velocity

INTEGER FLIP_ALF, FLIP_BET, FLIP_GAM, FLIP_SIN, FLIP_COS, &
& FLIP_V0, NUMBER_16

PARAMETER (FLIP_ALF=1, FLIP_BET=2, FLIP_GAM=3, FLIP_SIN=4, FLIP_COS=5, &
& FLIP_V0=6, NUMBER_16=7)

types 20.n = chopper (disk or blade); 6 parameters

.0 or .2 for motion in x-direction, .1 or .3 for vertical

.2 or .3 is counter-rotating (fully closed when edges at 0)

1) Linear velocity of opening crossing beam centerline (m/us)

2) Time to cover or uncover half the width of the moderator (us)

3) Nominal time at which opening chopper edge crosses zero (us)

4) Nominal time at which closing chopper edge crosses zero (us)

5) Phase jitter of chopper, rms (us)

6) Period of chopper (us)

INTEGER CHP_VEL, CHP_HALF, CHP_OPEN, CHP_CLOSE, &
& CHP_JITTER, CHP_PERIOD, NUMBER_20

PARAMETER (CHP_VEL=1, CHP_HALF=2, CHP_OPEN=3, CHP_CLOSE=4, &
& CHP_JITTER=5, CHP_PERIOD=6, NUMBER_20=7)

type 21.n = Fermi chopper; 14 or 18 parameters

.0, body transparent and blades perfectly opaque

.1, compute attenuation of body and blades

.2, blades are resonant absorbers (e.g. Cd or Gd)

1-3) (X0,Y0,Z0) location of center of rotation of the Fermi chopper (m)

4) Nominal time when chopper opening is parallel to Z-axis (us)

5) Rotation speed of Fermi chopper, positive if CCW (radian/us)

6) Phase jitter of chopper, rms (us)

7) Radius of chopper body (m)

8) Half the Height of slit package (m)

9) Half the Width of the slit-package insert including sagitta (m)

10) Half the Length of slit package (may be less than radius) (m)

11) Number of blades in slit package (1 more than # of channels)

12) Distance between blades (center-to-center) (m)

13) Thickness of absorbing part of blade (m)

14) Curvature (1/Radius of curvature) of blades (/m)

15-16) Attenuation length of body material, constant term (/m) and 1/v term (/us) (subtypes .1 or .2)

17-18) Attenuation length of blade material, constant term (/m) and 1/v term (/us) (subtype .1)

c 17-..) 7 followed by type 7 parameter block (subtype .2)

INTEGER FERMI_X0, FERMI_Y0, FERMI_Z0, FERMI_T0, F_OMEGA, F_JITTER, &
& FERMI_R, FERMI_H, FERMI_W, FERMI_L, F_NBLADE, F_DBLADE, &
& F_TBLADE, F_CURVE, F_BODY_0, F_BODY_V, F_BLADE_0, &
& F_BLADE_V, NUMBER_21

PARAMETER (FERMI_X0=1, FERMI_Y0=2, FERMI_Z0=3, FERMI_T0=4, F_OMEGA=5, &
& F_JITTER=6, FERMI_R=7, FERMI_H=8, FERMI_W=9, FERMI_L=10, &
& F_NBLADE=11, F_DBLADE=12, F_TBLADE=13, F_CURVE=14, &
& F_BODY_0=15, F_BODY_V=16, F_BLADE_0=17, F_BLADE_V=18, &
& NUMBER_21=19)
C type 22 = gravity focuser; 5 parameters
C 1,2) acceleration (m/us^2), and rms phase jitter (us)
C 3,4) nominal times for start and top of upward stroke (us)
C 5) time between pulses (us)
INTEGER G_ACCEL, G_JITTER, G_START, G_TOP, G_PERIOD, NUMBER_22
PARAMETER (G_ACCEL=1, G_JITTER=2, G_START=3, G_TOP=4, G_PERIOD=5, &
& NUMBER_22=6)
C type 23 = removable beamstop; no additional parameters
INTEGER NUMBER_23
PARAMETER (NUMBER_23=1)
C Class: samples
C 1) for most samples, macroscopic cross section at 1 A (/m)
C 2,3) samples may also have limited azimuthal range, min and max (radians)
INTEGER SIGMA_1A, PHI_MIN, PHI_MAX
PARAMETER (SIGMA_1A=1, PHI_MIN=2, PHI_MAX=3)
C type 30.n = small-angle scattering sample; 4 parameters
C 4) .0, fixed value of Q for scatter (1/A), 2 pi azimuthal angle
C .1, .3, or .5, biased azimuth in range [PHI_MIN, PHI_MAX]
C 4) .2 or .3, hard-sphere radius for scatter (A)
C 4) .4 or .5, minimum Q (1/A), scatter ~ 1/Q^2
INTEGER Q_SCATTER, R_SPHERE, NUMBER_30
PARAMETER (Q_SCATTER=4, R_SPHERE=4, NUMBER_30=5)
C type 32.n = isotropic scatterer with spectrum of discrete energy changes;
C .0 = 4 pi solid angle, single energy; 1 additional parameter
C .1,.3,.5 = limited solid angle; 8 additional parameters
C .2 or .3 = multi-level; 8 + 2*N additional parameters
C .4 or .5 = multi-level with widths; 8 + 3*N additional parameters
C .6 = multiple elastic scatter with absorption correction
C 2) inelastic neutron energy change (0 if elastic) (meV)
C 2) number of energy levels (if subtype 2 - 5)
C 2) ratio of scatter/total cross sections (subtype 6)
C 3,4) min and max of direction cosines with respect to X-axis
C 5,6) min and max of direction cosines with respect to Y-axis
C 7,8) min and max of direction cosines with respect to Z-axis
C 9) fraction of solid angle included, to modify statistical weight
C 10+) table of inelastic energy levels (meV); followed by table of
C cumulative probabilities ending with explicit 1; followed by
C widths (meV), >0 for Gaussian or <0 for Lorentzian
INTEGER DELTA_E, N_E_LEVELS, SCATRATIO, COS_X_MIN, COS_X_MAX, &
& COS_Y_MIN, COS_Y_MAX, COS_Z_MIN, COS_Z_MAX, &
& OMEGA_RATIO, E_LEVEL, NUMBER_32
PARAMETER (DELTA_E=2, N_E_LEVELS=2, SCATRATIO=2, COS_X_MIN=3, &
& COS_X_MAX=4, COS_Y_MIN=5, COS_Y_MAX=6, COS_Z_MIN=7, &
& COS_Z_MAX=8, OMEGA_RATIO=9, E_LEVEL=10, NUMBER_32=10)
C type 33.n = deep inelastic neutron scatter (DINS), 7 + 9*N parameters
C .0 = random azimuth in 2 pi
C .1 = biased azimuth in range [PHI_MIN, PHI_MAX]
C 1) total macroscopic hi-E cross section (NOT 1/v) (/m)
C 4) min Q to be sampled from 1/Q distribution (/A)
C 5) fraction of total cross section that is DINS, remainder absorbed
C 6) N, number of atomic species in sample (at least 1)
C 7+9i) normalized cumulative probability for species i, 0 <= i <= N-1
C 8+9i) atomic mass for atomic species i, 0 <= i <= N-1
C 9+9i) standard deviation of Gaussian for atomic species i (/A)
C 10+9i) A0,A1,A2,A3,A4 = expansion coefficient of the Compton profile,
C - 14+9i) in powers of x^2, for atomic species i
C 15+9i) final-state effect coefficient for atomic species i (/A)
INTEGER DINSQMIN, DINSFRAC, N_ATOMS, DINSPROB, DINS_M, DINS_SIG, &
PARAMETER (DINSQMIN=4, DINSFRAC=5, N_ATOMS=6, DINSPROB=7, DINS_M=8, &
DINS_SIG=9, DINS_A0=10, DINS_A1=11, DINS_A2=12, DINS_A3=13, &
DINS_A4=14, DINS_fse=15, NUMBER_33=16)

C type 34.n = inelastic scattering using S(a,b) file; 3 parameters, and NAME
C must be '[path]filename' for the file
C .0 = random azimuth in 2 pi
C .1 = biased azimuth in range [PHI_MIN, PHI_MAX]
C
INTEGER DENSITY, NUMBER_34
PARAMETER (DENSITY=1, NUMBER_34=4)

C type 35 = scattering from layered reflectometry sample; 1 + 4*N parameters
C 1) number of layers, including substrate
C parameters for each layer, starting with substrate:
C 2,3) 4pi*Real and Imaginary scattering-length density (1/A^2)
C 4) Thickness of layer (zero for substrate) (A)
C 5) Roughness, 2*sigma^2 of outer surface of this layer (A^2)
INTEGER NLAYERS, REAL4PINB, IMAG4PINB, THK_LAYER, RUF_LAYER, &
NUMBER_35
PARAMETER (NLAYERS=1, REAL4PINB=2, IMAG4PINB=3, &
THK_LAYER=4, RUF_LAYER=5, NUMBER_35=2)

C type 36.n = scattering from isotropic polycrystalline powder; 9+2*N parameters
C .0 = random azimuth in 2 pi
C .1 = biased azimuth in range [PHI_MIN, PHI_MAX]
C 4) number of Bragg edges included
C 5) limiting (short wavelength) macroscopic total xsection (1/cm)
C 6) macroscopic incoherent scattering xsection (1/cm)
C 7) macroscopic 1/v scattering xsection at 1 A, generally 0 (1/cm/A)
C 8) macroscopic 1/v absorption xsection at 1 A (1/cm/A)
C 9+) table of d-spacings of Bragg edges (A), followed by explicit 0 and
C table of cumulative macroscopic xsections at 1 A (1/cm/A^2)
INTEGER N_BRAGG, PSIGMAT, PSIGMAI, PSIGMAS, PSIGMAA, D_BRAGG, &
NUMBER_36
PARAMETER (N_BRAGG=4, PSIGMAT=5, PSIGMAI=6, PSIGMAS=7, PSIGMAA=8, &
D_BRAGG=9, NUMBER_36=10)

C type 37.n = scattering from single crystal sample; NAME must be
C [path]filename for file with h,k,l,|F|^2
C 1) absorption cross section at 2200 m/s (b)
C 2) incoherent scattering cross section (b)
C 3) number of (hkl) records to be read from file
C 4-6) first unit-cell vector, (a,0,0) if orthorhombic & normal to beam (A)
C 7-9) second unit-cell vector, (0,b,0) if orthorhombic & normal to beam (A)
C 10-12) third unit-cell vector, (0,0,c) if orthorhombic & normal to beam (A)
C 13) delta-d/d (for finite size effect), Gaussian rms
C 14) isotropic mosaic, Gaussian rms (leave 0 for anisotropic) ('')
C 15-23) anisotropic mosaic matrix, Gaussian rms ('')
INTEGER SC_abs, SC_inc, SC_count, SC_ax, SC_ay, SC_az, SC_bx, &
SC_by, SC_bz, SC_cx, SC_cy, SC_cz, SC_delta_d_d, &
SC_mosaic, NUMBER_37
PARAMETER (SC_abs=1, SC_inc=2, SC_count=3, SC_ax=4, SC_ay=5, SC_az=6, &
SC_bx=7, SC_by=8, SC_bz=9, SC_cx=10, SC_cy=11, SC_cz=12, &
SC_delta_d_d=13, SC_mosaic=14, NUMBER_37=15)

C class: detector
C NOTE: any detector with subtype .4 also records polarization vector
C 1) 1st parameter of all detectors is detector index number
C 2) 2nd parameter (types 40-44)is -ln(1 - efficiency at 1 A);
C set negative for "monitor" with 100% efficiency but no absorption.
C Time-Of-Flight clock parameters:
C 3,4) minimum and maximum times (us)
C 5) number of time channels
C 6) if logarithmic, dt/t (otherwise dt/t = 0)
C 7) minimum clock tick in determining log scale (us)
C 8) electronic delay of detector events, offset of log scale (us)
C 9) repeat period of data-acquisition electronics (us)
INTEGER DET_INDEX, D_ALPHA_1A, D_TMIN, D_TMAX, D_TCHANS, &
& D_DT_OVER_T, D_TICK, D_DELAY, D_T_PERIOD
PARAMETER (DET_INDEX=1, D_ALPHA_1A=2, D_TMIN=3, D_TMAX=4, D_TCHANS=5, &
& D_DT_OVER_T=6, D_TICK=7, D_DELAY=8, D_T_PERIOD=9)
C type 40 = detector; 9 additional parameters
INTEGER NUMBER_40
PARAMETER (NUMBER_40=10)
C type 41 = vertical linear detector; 14 additional parameters
C 10,11)locations of bottom and top of detector (m)
C 12) number of detector elements
C 13) size of detector element (m)
C 14) root-mean-square encoding error of detector (m)
INTEGER DET_YMIN, DET_YMAX, DET_NY, DET_DELY, DET_RMSY, NUMBER_41
PARAMETER (DET_YMIN=10, DET_YMAX=11, DET_NY=12, DET_DELY=13, &
& DET_RMSY=14, NUMBER_41=15)
C type 42 = horizontal linear detector; 14 additional parameters
C 10,11)locations of left and right ends of detector (m)
C 12) number of detector elements
C 13) size of detector element (m)
C 14) root-mean-square encoding error of detector (m)
INTEGER DET_XMIN, DET_XMAX, DET_NX, DET_DELX, DET_RMSX, NUMBER_42
PARAMETER (DET_XMIN=10, DET_XMAX=11, DET_NX=12, DET_DELX=13, &
& DET_RMSX=14, NUMBER_42=15)
C type 43.nm = 2-D detector; 22 additional parameters
C n = axis orientation digit: .0 = Z, .1 = X, .2 = Y
C (add .4 to n for recording polarization vector)
C m = coordinate type digit: .00 = rectilinear, .01 = plane polar,
C .02 = cylindrical, .03 = spherical
C
<table>
<thead>
<tr>
<th>m</th>
<th>n</th>
<th>.0, .4</th>
<th>.1, .5</th>
<th>.2, .6</th>
</tr>
</thead>
<tbody>
<tr>
<td>.00</td>
<td>(X, Y)</td>
<td>(Z, Y)</td>
<td>(X, Z)</td>
<td></td>
</tr>
<tr>
<td>.01</td>
<td>(rhoXY, phiXY)</td>
<td>(rhoZY, phiZY)</td>
<td>(rhoXZ, phiXZ)</td>
<td></td>
</tr>
<tr>
<td>.02</td>
<td>(Z-Z0, phiXY)</td>
<td>(X-X0, phiZY)</td>
<td>(Y-Y0, phiXZ)</td>
<td></td>
</tr>
<tr>
<td>.03</td>
<td>(thetaZ, phiXY)</td>
<td>(thetaX, phiZY)</td>
<td>(thetaY, phiXZ)</td>
<td></td>
</tr>
</tbody>
</table>

where: rhoXY = sqrt[(X-X0)^2 + (Y-Y0)^2)] = distance from Z-axis (m),
phiXY = arctan[(Y-Y0)/(X-X0)] = angle from +X toward +Y (rad),
thetaZ = arctan[rhoXY/(Z-Z0)] = angle away from Z-axis (rad)
C 10,11)locations of min and max X (abscissa) edges of detector (m or rad)
C 12) number of detector elements in the abscissa direction
C 13) width of detector abscissa element (m or rad)
C 14) root-mean-square abscissa encoding error of detector (m or rad)
C 15,16)locations of min and max Y (ordinate) edges of detector (m or rad)
C 17) number of detector elements in the ordinate direction
C 18) height of detector ordinate element (m or rad)
C 19) root-mean-square ordinate encoding error of detector (m or rad)
C 20-22)origin of coordinates for curved surfaces (if any) (m)
INTEGER DET2_XMIN, DET2_XMAX, DET2_NX, DET2_DELX, DET2_RMSX,
& DET2_YMIN, DET2_YMAX, DET2_NY, DET2_DELY, DET2_RMSY, &
& DET2_X0, DET2_Y0, DET2_Z0, NUMBER_43
PARAMETER (DET2_XMIN=10, DET2_XMAX=11, DET2_NX=12, DET2_DELX=13, &
& DET2_RMSX=14, DET2_YMIN=15, DET2_YMAX=16, DET2_NY=17, &
& DET2_DELY=18, DET2_RMSY=19, DET2_X0=20, DET2_Y0=21, &
& DET2_Z0=22, NUMBER_43=23)
C type 44 = longitudinal linear detector; 14 additional parameters
C 10,11) locations of upstream and downstream ends of detector (m)
C 12) number of detector elements
C 13) size of detector element (m)
C 14) root-mean-square encoding error of detector (m)
INTEGER DET_ZMIN, DET_ZMAX, DET_NZ, DET_DELZ, DET_RMSZ, NUMBER_44
PARAMETER (DET_ZMIN=10, DET_ZMAX=11, DET_NZ=12, DET_DELZ=13, &
& DET_RMSZ=14, NUMBER_44=15)
C types 45-49 = Resonance detectors, corresponding to geometries of 40-44.
C 2) thickness of resonant foil, + for capture or - for absorption (m)
C Parameters must be followed immediately by a type 7 material; that
C is, NUMBER_4x will be used as a pointer
C NUMBER_4x) Doppler velocity width (m/us)
C NUMBER_4x+1) pointer to the material region.
INTEGER RESDET_dX
PARAMETER (RESDET_dX=2)

C type 50 = scattering chamber, void-filled. No parameters, but other regions
C may be embedded, indicated by surface types with 10s digit on.
C NOTE: a special use of a chamber provides for rotation of the beam axis in
C the horizontal plane on exit if the surface type is +/- 5.
C In that case there are 4 (unnamed) parameters:
C 1-2) X,Z of axis of rotation (vertical axis) (m)
C 3-4) sine and cosine of rotation angle
INTEGER NUMBER_50
PARAMETER (NUMBER_50=1)
C type 51.n = wrapper for regions with polarization dependence. No parameters.
C .0 = select one spin, .1 = split and track both spins.
C Following two regions must have identical geometry, for spin parallel and
C spin antiparallel to magnetic field respectively.
INTEGER NUMBER_51
PARAMETER (NUMBER_51=1)

C class: magnetic field regions represented by induction vector B
C 1) first parameter of all B regions may be pulsed-source repetition
C time, or zero for a steady field (us)
C 2) pulse reference time T0, such that B(t) ~ 1/(t-T0) (us)
C 3) initial field time T1 measured from T0; B(T0+T1) = B from
C various algorithms and B(t) = B*T1/(t-T0) (us)
C 4) latest field time T2 measured from T0 (us)
INTEGER B_PERIOD, B_T0, B_T1, B_T2
PARAMETER (B_PERIOD=1, B_T0=2, B_T1=3, B_T2=4)
C type 60 = uniform field; 8 parameters
C 5-7) (X,Y,Z) components of magnetic induction, B1 (T)
C 8) |B1| (T)
INTEGER B1x, B1y, B1z, ABSB_1, NUMBER_60
PARAMETER (B1x=5, B1y=6, B1z=7, ABSB_1=8, NUMBER_60=9)
C type 61.n = gradient field between 2 surfaces; 39 parameters
C .0 = linear interpolation of components of B
C .1 = helical (linear magnitude and angle)
C 5-7) (X,Y,Z) components of magnetic induction at surface 1, B1 (T)
C 8) |B1| (T)
C 9-11) (X,Y,Z) components of magnetic induction at surface 2, B2 (T)
C 12) |B2| (T)
C 13-22) coefficients of surface 1, normalized to give distance in m
C 23-32) coefficients of surface 2, normalized to give distance in m
C 33-38) matrix elements for helical interpolation (/T)
C 39) angle between B1 and B2 vectors (rad)

INTEGER B2x, B2y, B2z, ABSB_2, SURF1, SURF2, Bhelix, B1thetaB2, &
& NUMBER_61
PARAMETER (B2x=9,B2y=10,B2z=11,ABSB_2=12,SURF1=13,SURF2=23, &
& Bhelix=33,B1thetaB2=39,NUMBER_61=40)

C type 62 = superposition of current loops; 6 + 8*N parameters
C 5) number of current loops
C 6) number of parameters to describe each loop (8)
C 7-9) X,Y,Z of center of 1st loop (m)
C 10-12) direction cosines of 1st loop axis
C 13) radius of 1st loop (m)
C 14) current in 1st loop (A)
C 15-22) parameters of 2nd loop, etc.

INTEGER N_LOOPS, LOOPN, LOOPX, LOOPY, LOOPZ, LOOPALF, LOOPBET, &
& LOOPGAM, LOOPRAD, LOOPCRNT, NUMBER_62
PARAMETER (N_LOOPS=5,LOOPN=6,LOOPX=7,LOOPY=8,LOOPZ=9,LOOPALF=10, &
& LOOPBET=11,LOOPGAM=12,LOOPRAD=13,LOOPCRNT=14, &
& NUMBER_62=7)

C type 63 = solenoid; 13 parameters
C 5-7) X,Y,Z at center of the solenoid (m)
C 8-10) direction cosines of the solenoid axis
C 11) radius of the solenoid (m)
C 12) half of the length of the solenoid (m)
C 13) surface current density, turns*current/length (A/m)

INTEGER SOL_XCENT, SOL_YCENT, SOL_ZCENT, SOL_ALF, SOL_BET, &
& SOL_GAM, SOL_R, SOL_HL, SOL_J, NUMBER_63
PARAMETER (SOL_XCENT=5,SOL_YCENT=6,SOL_ZCENT=7,SOL_ALF=8, &
& SOL_BET=9,SOL_GAM=10,SOL_R=11,SOL_HL=12,SOL_J=13, &
& NUMBER_63=14)

C type 64 = finite coaxial alternating current sheets; 16 parameters
C 5) number of sheets
C 6) spacing of sheets (m)
C 7-9) X,Y,Z of center of 1st sheet (m)
C 10-12) direction cosines of sheet normal (axis of array)
C 13-14) transverse dimensions of sheets (m)
C 15) sheet current (A)
C 16) azimuthal angle of current in 1st sheet (rad)

INTEGER N_SHEETS, SHT_SPACE, SHT_XCENT, SHT_YCENT, SHT_ZCENT, &
& SHT_ALF, SHT_BET, SHT_GAM, SHT_DX, SHT_DY, SHT_I, &
& SHT_PHI, NUMBER_64
PARAMETER (N_SHEETS=5,SHT_SPACE=6,SHT_XCENT=7,SHT_YCENT=8, &
& SHT_ZCENT=9,SHT_ALF=10,SHT_BET=11,SHT_GAM=12, &
& SHT_DX=13,SHT_DY=14,SHT_I=15,SHT_PHI=16,NUMBER_64=17)

C type 65 = field defined in file; 4 parameters

INTEGER NUMBER_65
PARAMETER (NUMBER_65=5)

C type 66 = multiple connected wire segments; 9+3*N parameters
C 5) number of nodes = 1 + number of wires
C 6) current, in direction from node 1 to node 2, etc. (A)
C 7-9) (X1,Y1,Z1) at beginning of first segment (m)
C 10...) (X1,Y1,Z1) at successive nodes (m)

INTEGER W_NODES, W_CURRENT, W_X1, W_Y1, W_Z1, NUMBER_66
PARAMETER (W_NODES=5,W_CURRENT=6,W_X1=7,W_Y1=8,W_Z1=9, &
& NUMBER_66=10)

C type 69 = superposition of any number of field types; 5+n parameters
C 5) number of superposed fields
C 6...) offset to parameter block with field definition
INTEGER N_FIELDS, B_OFFSET, NUMBER_69
PARAMETER (N_FIELDS=5, B_OFFSET=6, NUMBER_69=7)

C types 90.n = source size and phase space to be sampled; 14-18 parameters
C .1 = first aperture rectangular, .2 = second aperture rectangular,
C .4 = either/both offset vertically (options are additive)
C 1-4) Xmin, Xmax, Ymin, Ymax edges of rectangular moderator face (m)
C 5,6) location and radius (half-width) of 1st beam-defining aperture (m)
C 7,8) location and radius (half-width) of 2nd beam-defining aperture (m)
C 9) additional vertical space to sample for gravity focus (m)
C 10,11) min and max neutron energy to be sampled (meV)
C 12,13) time between beam pulses, and proton pulse width (us)
C 14) offset to parameter block with spectrum and lineshape parameters
C 15,16) (optional) half-heights of apertures, type 90.1 or 90.2 (m)
C 17,18) (optional) vertical offsets of apertures, type 90.4 (m)

INTEGER MOD_XMIN, MOD_XMAX, MOD_YMIN, MOD_YMAX, APTR1_Z, APTR1_R, &
& APTR2_Z, APTR2_R, G_DELY2, S_EMIN, S_EMAX, S_PERIOD, &
& S_WIDTH, E_OFFSET, APTR1_Y, APTR2_Y, APTR1_Y_OFFSET, &
& APTR2_Y_OFFSET, NUMBER_90
PARAMETER (MOD_XMIN=1, MOD_XMAX=2, MOD_YMIN=3, MOD_YMAX=4, APTR1_Z=5, &
& APTR1_R=6, APTR2_Z=7, APTR2_R=8, G_DELY2=9, S_EMIN=10, &
& S_EMAX=11, S_PERIOD=12, S_WIDTH=13, E_OFFSET=14, &
& APTR1_Y=15, APTR2_Y=16, APTR1_Y_OFFSET=17, &
& APTR2_Y_OFFSET=18, NUMBER_90=19)

C type 91 = source energy distribution table and lineshape parameters; 12
C parameters plus length of table
C 91.1 special case for polarized fixed velocity source (see param 13)
C 1) number of entries in energy table (1 for special cases)
C 2) location in table of center of normal distribution (index units);
C OR value of nominal neutron velocity (m/us) (if # entries = 1);
C OR 0 (and # entries = 1) as flag to use Maxwellian distribution(s)
C 3) standard deviation of normal distribution (table index units);
C OR (if # entries = 1) + relative fwhm of triangular or - relative
C full width of rectangular velocity distribution;
C OR (if Maxwellian flag) number of Maxwellians to sum
C 4) source brightness, summed over limits of E_TABLE (n/ster/m^2/MW/s)
C 5) 0 for no time distribution, <0 for spline tables (see type 92), or
C >0 for one of the following parametrizations, chosen by signs of
C coefficients (7) & (10):
C Gaussian * exponential: (7) >= 0, (10) >=0
C Ikeda-Carpenter * exponential: (7) >= 0, (10) < 0
C Ikeda-Carpenter, NO exponential: (7) >= 0, (10) = -1
C Parameterized Ikeda-Carpenter: (7) < 0, (10) < 0
C 11,12) switching function 1/e point (A), and power (slope)
C 13) origin of table of cumulative energy distribution (weighted by
C lambda^2) of source spectrum on equally spaced normal-curve values
C of log(energy/1meV)
C 13-15) special case for type 91.1 and # entries = 1, fixed polarization
C components for all source particles: PX, PY, and PZ
C 13-..) special case for sum of Maxwells: cumulative probability
C and Maxwellian KT (meV) pairs
INTEGER N_E_TABLE, CENT_TABLE, SIGMA_TABLE, S_BRIGHT, TAU_TH1, &
  TAU_TH2, TAU2_RATIO, TAU_EPI, T_DELAY, T_WIDTH, &
  SWITCH_LAMBDA, SWITCH_POWER, E_TABLE, NUMBER_91
PARAMETER (N_E_TABLE=1, CENT_TABLE=2, SIGMA_TABLE=3, S_BRIGHT=4, &
  TAU_TH1=5, TAU_TH2=6, TAU2_RATIO=7, TAU_EPI=8, T_DELAY=9, &
  T_WIDTH=10, SWITCH_LAMBDA=11, SWITCH_POWER=12, E_TABLE=13, &
  NUMBER_91=13)
C type 92 = source energy AND time distribution tables using cubic splines.
C 1) number of entries in combined tables (always 198)
C 4) source brightness, summed over limits of E_TABLE (n/ster/m^2/MW/s)
C 5) - number of spline nodes (always -33)
C 8,9) knee of switch function: 0<X<1 and 0<R<1
C 10) mean value of wavelength in the lambda-weighted distribution (A)
C 11,12) trapezoidal switching function min and max lambda (A)
C 13) origin of table of log10(cumulative lambda x energy distribution)
INTEGER SW_KNEE_X, SW_KNEE_R, LAMBDA_BAR, SW_MIN_LAMBDA, &
  SW_MAX_LAMBDA, NUMBER_92
PARAMETER (SW_KNEE_X=8, SW_KNEE_R=9, LAMBDA_BAR=10, SW_MIN_LAMBDA=11, &
  SW_MAX_LAMBDA=12, NUMBER_92=13)
C type 95 = source file, direct-access binary; no parameters, but NAME must
C be '[path]filename' for the file. File format is
C 1st record: 16 bytes of coding information and 17-character ID
C 2nd record: 40-character TITLE of job that created the file
C 3rd record: source surface definition
C 4th record: # of histories (integer*4), sum of weights, source MW-s,
C modified source brightness, phase space, lethargy
C records 5-# of histories+4): source neutrons
INTEGER NUMBER_95
PARAMETER (NUMBER_95=1)
C********** INCLUDE FILE, 'MC_GEOM.INC' **********
C 08 Feb 1994: converted from COMMON to STRUCTURES. Added WT, M, and POL to
C particle record. Require Y direction to be vertical (for
C gravity). [PAS]
C 17 Feb 1994: added BETA to surface record; type 5 boundary; made MAXP
C a parameter [PAS]
C 15 Feb 1995: include structure /MC_ELEMENT/ formerly in MC_ELEMNT.INC [PAS]
C 04 Dec 1995: larger MAXR, MAXP; replace POL with Q and (PX,PY,PZ) [PAS]
C 27 Nov 1997: larger MAXS, MAXR, MAXP (total size 65530 bytes) [LLD,PAS]
C 14 Sep 1998: SURFACE and PARAM changed from REAL*4 to REAL*8 [PAS]
C 26 Mar 1999: larger MAXS, MAXR, MAXP (total size 121648 bytes) [Fitz,PAS]
C 30 Sep 2003: larger MAXS, MAXR, MAXP (total size 227722 bytes) [PAS]
C 14 Jun 2005: larger MAXS (total size 31432 bytes) [PAS]
C 07 Jul 2006: make MAXM (number of magnetic regions) a parameter [PAS]
C To change dimensions of all arrays, change maximum numbers of
C surfaces (MAXS), regions (MAXR), element parameters (MAXP), and/or
C magnetic regions (MAXM) in following PARAMETER statement.
INTEGER*4 MAXS, MAXR, MAXP, MAXM
PARAMETER (MAXS=400, MAXR=255, MAXP=8000, MAXM=50)
C Definitions of surface parameters and array of boundaries of surfaces:
C SURFACE is a record containing 10 coefficients of a general
C quadratic surface, of the form
C A*(X**2) + B*X + C*(Y**2) + D*Y + E*(Z**2) + F*Z + G +
MCLIB  A-13

\[ P^*(X*Y) + Q^*(Y*Z) + R^*(Z*X) = 0 \]

**BETA** = surface roughness parameter between 0 (smooth) and 1 (cosine);

negative or >1 is completely random

**STRUCTURE** /SURFACE/

REAL*8 A, B, C, D, E, F, G, P, Q, R
REAL*8 BETA

END STRUCTURE

**REGION** is record containing NSURF values of the I*2 variable IGEOM,

defining a region by its bounding surfaces:

+  if interior of region is on + side of surface
-
  if interior of region is on - side of surface
0  if surface is not a boundary of the region
1  for ordinary surface described by roughness BETA and possibility
of refraction or critical reflection
2  for totally reflecting surface
3  for diffuse scattering surface
4  for absorbing surface
5  special action required in previous region before crossing surface
6  split particle by 2 after crossing surface, treated as type 1
10  for boundary of a magnetic field region
+10  for a surface which bounds an embedded region within the region; not
    tested to determine if within region, but tested for exit from region
+20  for embedded surface with possible regions on BOTH sides

**STRUCTURE** /REGION/

INTEGER*2 IGEOM(MAXS)

END STRUCTURE

**NSURF** = number of surfaces defined
**NREG** = number of regions defined

**STRUCTURE** /MC_GEOM/

INTEGER*4 NSURF,NREG

RECORD /SURFACE/ SURFACE(MAXS)
RECORD /REGION/ REGION(MAXR)

END STRUCTURE

Definitions of element-parameter block

Each INDEX points to the beginning of a structure within the contiguous
block **PARAM**. The first entry in **PARAM** for each element identifies the
type, followed by a varying number of parameters.
**NAME** = 40-character descriptive name of a region or element
**NEXTINDX** = pointer to next available location in **PARAM**
**INDEX** = pointer into parameter block; if 0, region is not an "element"
**PARAM** = block of element parameters

**STRUCTURE** /MC_ELEMENT/

CHARACTER*40 NAME(MAXR)
INTEGER*4  NEXTINDX, INDEX(MAXR)
REAL*8  PARAM(MAXP)

END STRUCTURE

Definitions of position and velocity of particle:

\((X, Y, Z)\) = position of particle (m); note that +Y is UP (for gravity)
\((VX, VY, VZ)\) = velocity of particle (m/us)
**TOF** = time of flight of particle (us)
**M** = atomic number of particle (e.g., 1 for neutron, 0 for photon)
**Q** = charge number of particle (e.g., 0 for neutron, +1 for proton)
**WT** = statistical weight of particle
**PX, PY, PZ** = average polarization vector, from -1.0 to +1.0

**STRUCTURE** /PARTICLE/

REAL*4 X, Y, Z, VX, VY, VZ, TOF
REAL*4 M, Q, WT, PX, PY, PZ
END STRUCTURE

C******** INCLUDE FILE, 'CONSTANT.INC' **********

C Various physical constants
C 21 Dec 1999: changed definition and units of PRECES_N
C 25 Jul 2006: added Boltzmann's constant, ROOT12
C
C BOLTZMN = Boltzmann gas constant (meV/K)
C GOVER2 = half the acceleration of gravity (m/us**2)
C HOVERM = Plank's constant/neutron mass (m-A/us)
C HSQOV2M = Plank's constant squared over 2 Mneutron (meV-A**2)
C PRECES_N = neutron magnetic-moment precession rate factor
C = - gamma = 2 Mun/(h/2pi) (radian/T/us)
C ROOTM_2 = square root of half the neutron mass (meV**0.5-us/m)
C ROOT12 = sqrt(12)
C TWOPI = 2 pi
C
REAL*8 BOLTZMN, GOVER2, HOVERM, HSQOV2M, PRECES_N, ROOTM_2,
& ROOT12, TWOPI
PARAMETER (BOLTZMN = 0.08617342,
& GOVER2 = 4.858d-12,
& HOVERM = 0.0039560339,
& HSQOV2M = 81.8145347,
& PRECES_N = 183.24702,
& ROOTM_2 = 2286.26,
& ROOT12 = 3.46410161513775d0,
& TWOPI = 6.28318530717959d0)
### Subroutine Abstracts

#### C++

`**********    A L O O P    **********`

```c
REAL*8 FUNCTION ALOOP(X, Y, Z, PARAMS, A)
```

Magnetic vector potential (phi component clockwise about loop axis) at point (X,Y,Z) produced by one current loop with parameters given in PARAMS. The vector potential in the lab coordinate system is also returned as the 3-vector A. The algorithm is from E. Durand, "Magnetostatique," (Masson, Paris, 1968) pp. ???. In the coordinate system of the loop, A(rho,z) = factor*[...], where G(rho,z;zeta) = 2 a (z-zeta)/R ([E(k)-K(k)])/k^2 - (1+n)/n [PI(n,k)-K(k)]

and R=sqrt((a+rho)^2+(z-zeta)^2), k=2 sqrt(a rho)/R, n=-4 a rho/(a+rho)^2 and K(k),E(k) are complete elliptic integrals of 1st & 2nd kinds.

P.A. Seeger, March 10, 2002

Variables in calling sequence:
- X, Y, Z = coordinates where the induction is to be calculated (m) (input)
- PARAMS = parameters of the loop: X (m), Y (m), Z (m), alpha, beta, gamma, radius (m), and current (A) or moment/pi (A*m^2) (input)
- A = vector potential in (left-handed) lab coordinates (T-m) (output)

IMPLICIT NONE

```
#include  'constant.inc'
```

```
REAL*8   X, Y, Z, PARAMS(8), A(3)
```

No Externals:

#### C++

`**********    A N G L I    **********`

```c
SUBROUTINE ANGLI(PART, SURF, COSTH, AP, BP, CP)
```

Compute the cosine of the angle of incidence of the particle PART with respect to surface SURF. The cosine is returned in COSTH, and the direction cosines of the normal to the surface as AP, BP, CP.

No test is made to assure that particle is actually on the surface.


03 Feb 1994: converted from COMMON to STRUCTUREs; changed calling sequence [PAS]

14 Sep 1998: all REAL*4 changed to REAL*8; DSQRT [PAS]

Definitions of STRUCTUREs:

```c
IMPLICIT NONE
INCLUDE 'mc_geom.inc'
```

Variables in calling sequence:
- PART = record containing particle coordinates (input)
- SURF = record containing definition of surface (input)
- COSTH = cosine of angle between /MC_PART/ and surface normal (output)
- AP, BP, CP = direction cosines of surface normal (output)

```
record /particle/ part
record /surface/ surf
```
REAL*8 COSTH, AP, BP, CP
C
C Externals
C NORM8
C--
C++
C********** A N G T O R U S **********
C
SUBROUTINE ANGTORUS(PART, PARAM, COSTH, AP, BP, CP, ISEED)
C
Compute the cosine of the angle of incidence of the particle PART with
C respect to a toroidal surface defined in PARAM, which may include wobble.
The cosine is returned in COSTH, and the direction cosines of the normal
to the surface as AP,BP,CP. No test is made to assure that particle is
C actually on the torus.
C
From subroutines ANGLI and WOBBLE, P. A. Seeger, August 4, 1996.
C 14 Sep 1998: all REAL*4 changed to REAL*8; DSQRT [PAS]
C 22 Jan 1999: make DX,DY,DZ REAL*4 for call to ORRAND [PAS]
C 20 Aug 2003: allow inner half-shell as well as outer [PAS]
C
Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'
INCLUDE 'mc_elmnt.inc'
C
Variables in calling sequence:
C PART = record containing particle coordinates (input)
C PARAM = block containing definition of toroidal surface (input)
C COSTH = cosine of angle between PART and surface normal (output)
C AP,BP,CP = direction cosines of surface normal (output)
C ISEED = random-number generator seed (input/output)
RECORD /PARTICLE/ PART
REAL*8 PARAM(0:*), COSTH, AP, BP, CP
INTEGER*4 ISEED
C
Externals:
C ORRAND
C--
C++
C**********   A S O L N D   **********
C
REAL*8 FUNCTION ASOLND(X, Y, Z, PARAMS, A)
C
Magnetic vector potential (phi component clockwise about loop axis) at point
C (X,Y,Z) produced by a solenoid with parameters given in PARAMS. The vector
C potential in the lab coordinate system is also returned as the 3-vector A.
The algorithm is from E.Durand, "Magnetostatique," (Masson, Paris, 1968)
C pp.97-100. In the coordinate system of the solenoid,
C A(rho,z) = factor*[G(rho,z;-b) - G(rho,z;b)],
C where G(rho,z;zeta) = 2 a (z-zeta)/R ([E(k)-K(k)]/k^2 - (1+n)/n [PI(n,k)-
C K(k)])
C and R=sqrt((a+rho)^2+(z-zeta)^2), k=2 sqrt(a rho)/R, n=-4 a rho/(a+rho)^2
c and K(k),E(k),PI(n,k) are complete elliptic integrals of 1st, 2nd, & 3rd
kinds.
C
P.A.Seeger, January 09, 2002
C 10 Mar 2002: added vector A to calling sequence [PAS]
C
Variables in calling sequence
X,Y,Z = coordinates where the induction is to be calculated (m) (input)
PARAMS = parameters of the region containing the solenoid (input)
A = vector potential in (left-handed) lab coordinates (T-m) (output)

IMPLICIT NONE
INCLUDE 'constant.inc'
INCLUDE 'mc_elmnt.inc'
REAL*8 X, Y, Z, PARAMS(0:*), A(3)
C
C Externals:
C CMPLT_PI
REAL*8 CMPLT_PI
C--
C++
C********** A T T E N _ A l **********
C
REAL*8 FUNCTION ATTEN_AL(LAMBDA, TIN, ABSORB, TWOSINTH, ISEED)
C
Neutron attenuation length (mm) of polycrystalline aluminum at
temperature TIN (default 300K), as a function of neutron wavelength
LAMBDA in Angstroms. May also return the fraction of the cross
section which is nuclear absorption (ABSORB). May also be used to
scatter from a random selected plane spacing and return TWOSINTH.
A special case is TWOSINTH=2, indicating isotropy in C.M. system.

From J. M. Carpenter et al., SIGAL, January 16, 1986 (IPNS Note 32)
28 Jan 1993: adapted, restructured, modified numeric values [PAS]
30 Sep 2003: added optional parameter TIN to calling sequence;
updated parameters; added temperature dependence. [PAS]
09 Mar 2006: added optional ABSORB ratio to calling sequence [PAS]
09 May 2009: added optional TWOSINTH and ISEED, to select a random
d-spacing; changed variables to REAL*8; restructured [PAS]
18 May 2009: increased NMAX from 34 to 88; series expansion for
single-phonon contribution [PAS]
24 May 2009: change normalization to match SIGscat at hi-E [PAS]
24 Jun 2009: make test of same LAMBDA single-precision [PAS]
C
Variables in calling sequence:
LAMBDA = neutron wavelength (A) (input)
TIN = material temperature (K) (optional input)
ABSORB = ratio of absorption/total cross section (optional output)
TWOSINTH = twice the sine of Bragg angle (optional output)
ISEED = random number seed (optional in/out)
IMPLICIT NONE
INCLUDE 'constant.inc'
REAL*8 LAMBDA
REAL*8,OPTIONAL :: TIN, ABSORB, TWOSINTH
INTEGER*4,OPTIONAL :: ISEED
C
C Externals:
C HUNT8 RAN
REAL*4 RAN
C--
C++
C********** A T T E N _ B e **********
C
REAL*8 FUNCTION ATTEN_Be(LAMBDA, TIN, ABSORB, TWOSINTH, ISEED)
C
Neutron attenuation length (mm) of polycrystalline beryllium at
temperature TIN (optional, default 100K), as a function of neutron
wavelength LAMBDA in Angstroms. May also return the fraction of the cross section which is nuclear absorption (ABSORB). May also be used to scatter from a random selected plane spacing and return TWOSINTH. A special case is TWOSINTH=2, indicating isotropy in the C.M. system.

From J. M. Carpenter et al., SIGAL, January 16, 1986 (IPNS Note 32)
25 Jan 1995: converted from Al to Be [PAS & Bob VonDreele]
30 Sep 2003: added optional parameter TIN to calling sequence;
updated parameters; added temperature dependence [PAS]
09 Mar 2006: added optional ABSORB ratio to calling sequence [PAS]
18 May 2009: added optional TWOSINTH and ISEED, to select a random d-spacing; changed variables to REAL*8; restructured increased NMAX from 28 to 200; series expansion for single-phonon contribution [PAS]
24 May 2009: change normalization to match SIGscat at hi-E [PAS]
24 Jun 2009: make test of same LAMBDA single-precision [PAS]

Variables in calling sequence:
LAMBDA = neutron wavelength (A) (input)
TIN = material temperature (K) (optional input)
ABSORB = ratio of absorption/total cross section (optional output)
TWOSINTH = twice the sine of Bragg angle (optional output)
ISEED = random number seed (optional in/out)

IMPLICIT NONE
INCLUDE 'constant.inc'
REAL*8 LAMBDA
REAL*8,OPTIONAL :: TIN, ABSORB, TWOSINTH
INTEGER*4,OPTIONAL :: ISEED

EXTERNALS:

HUNT8 RAN
REAL*4 RAN

** A T T E N _ F e *******

REAL*8 FUNCTION ATTEN_Fe(LAMBDA, TIN, ABSORB, TWOSINTH, ISEED)

Neutron attenuation length (mm) of polycrystalline iron at temperature TIN (default 300K), as a function of neutron wavelength LAMBDA in Angstroms. May also return the fraction of the cross section which is nuclear absorption (ABSORB). May also be used to scatter from a random selected plane spacing and return TWOSINTH. A special case is TWOSINTH=2 indicating isotropy in C.M. system.

From J. M. Carpenter et al., SIGAL, January 16, 1986 (IPNS Note 32)
18 May 2009: copied from ATTEN_Al with different parameters [PAS]
24 May 2009: change normalization to match SIGscat at hi-E [PAS]
24 Jun 2009: make test of same LAMBDA single-precision [PAS]

Variables in calling sequence:
LAMBDA = neutron wavelength (A) (input)
TIN = material temperature (K) (optional input)
ABSORB = ratio of absorption/total cross section (optional output)
TWOSINTH = twice the sine of Bragg angle (optional output)
ISEED = random number seed (optional in/out)
IMPLICIT NONE
INCLUDE 'constant.inc'
REAL*8 LAMBD
REAL*8,OPTIONAL :: TIN, ABSORB, TWOSINTH
INTEGER*4,OPTIONAL :: ISEED

C Externals:
C   HUNT8   RAN
REAL*4 RAN
C--
C++
C********** A T T E N _ X **********
C
REAL*8 FUNCTION ATTEN_X(PARAMS, LAMBD)
C
C Neutron attenuation length (mm) of a single-crystal filter, using the
C formalism of Freund, as a function of neutron wavelength LAMBD:
C sigma = sigfree*[1-exp(-C/lambda**2)] + sigabs*lambda
C
C 14 Aug 1995: was returning reciprocal [PAS,GXu]
C 14 Sep 1998: all REAL*4 changed to REAL*8 [PAS]
C
C Variables in calling sequence:
C PARAMS = array with cross section parameters (input)
C LAMBD = neutron wavelength (A) (input)
IMPLICIT NONE
C
INCLUDE 'mc_elmnt.inc'
REAL*8 PARAMS(0:*), LAMBD
C
C No externals
C--
C++
C**********  B F I E L D  **********
C
SUBROUTINE BFIELD(PART, P, DIST, GEOM, MREG, NAME, PARAMS, B,     &
&                  NSTEPS, PRECES)
C                SETRKEPS(EPSIN)
C
C*** Must be compiled with array-bound checking turned OFF
C
C Find either the value of the magnetic induction 3-vector B at the location
C of PART, or if DIST is not 0 a simple interpolation of B in the middle of
C the path length. The parameters of region MREG are given in PARAMS, and
C the geometric description of MREG is available in structure GEOM. If the
C polarization vector P is not zero, the line integral of (-gamma|B|dt)
C over DIST is computed and returned as the precession angle, PRECES, and P
C is updated. For constant field direction NSTEPS is 0, otherwise it is the
C number of steps used to integrate Bloch's precession equation. NSTEPS
C will be negative if the integration was not done. To allow external
C adaptive stepsize, DIST may also be reduced. An entry point is provided to
C set the precision parameter for the Runge-Kutta routine.
C
C P.A. Seeger, 21 Dec 1999, includes constant and linear gradient regions.
C 26 Dec 1999: added type 62 (sum of current loops) [PAS]
C 27 Jan 2000: no computation of derivative in TYPE62_Broutine; PRECES in
C calling sequence; move Type 61 to subroutine [PAS]
C 31 Jan 2000: make auxiliary routines FUNCTIONS that return |B| [PAS]
C 02 Feb 2000: no FLAG in call; NSTEPS>=0; always return PRECES [PAS]
C 11 Feb 2000: added NAME to call, for look-up table files [PAS]
Definitions of STRUCTUREs:
  IMPLICIT NONE
  INCLUDE 'mc_geom.inc'
  INCLUDE 'mc_elmnt.inc'
  INCLUDE 'constant.inc'

Variables in calling sequence:
  PART = record with particle coordinates (input)
  P = polarization vector (input/output)
  DIST = distance to integrate B to find average (m) (input/output)
  GEOM = structure with definitions of all surfaces and regions (input)
  MREG = magnetic region particle is in (input)
  NAME = region or file name for look-up tables (input)
  PARAMS = array with description of region MREG (input/output)
  B = 3-vector with the field at a point, or average field (T) (output)
  NSTEPS = number of steps for precession integral, negative if integral
         not done (output)
  PRECES = line integral of -gamma |B| dt (rad) (output)
  EPSIN = precision limit for Runge-Kutta (input)

EXTERNAL
  FUNCTION TYPE60_Bfunction, TYPE61_Bfunction, TYPE62_Bfunction, &
         TYPE63_Bfunction, TYPE66_Bfunction, TYPE69_Bfunction
  TYPE60_Bfunction, TYPE61_Bfunction, TYPE62_Bfunction, &
  TYPE63_Bfunction, TYPE66_Bfunction, TYPE69_Bfunction

---

Use Romberg's method to integrate |B| from time T1 (us) to T2 (us) with
absolute error less than EPS, along the neutron trajectory given in PART,
using parameters found in PARAMS. Maximum number of points used in
extrapolation (KMAX) and maximum number of step divisions (JMAX) are set
in a PARAMETER statement.

IMPLICIT NONE
INTEGER JMAX, KMAX
PARAMETER (JMAX=10, KMAX=5)

From QROMB, Press, Teukolsky, Vettering, & Flannery, "Numerical Recipes,
Second Edition" (Cambridge University Press, 1992) sec. 4.3
Adapted specifically to |B|, put excerpts of TRAPZD and POLINT inline,
Variables in calling sequence:
- Bfunction = external function to compute |B| (input)
- T1 = time at beginning of integral (us) (input)
- T2 = time at end of integral (us) (input)
- EPS = allowed absolute error (T-us)
- PART = record with neutron trajectory for Bfunction (input)
- PARAMS = array with parameters for Bfunction (input)
- NAME = file name for look-up tables for Bfunction (input)

REAL*8    Bfunction, T1, T2, EPS, PART(*), PARAMS(*)
CHARACTER NAME*40
EXTERNAL  Bfunction

---

SUBROUTINE BLOOP(X, Y, Z, PARAMS, B)

Magnetic induction produced at point (X,Y,Z) by a single current loop whose
parameters are given in PARAMS. The induction is returned in the 3-vector B.
The algorithm for magnetic induction of a loop is from
The radius of the conducting wire is assumed to be 10 micron.
Complete elliptic integrals of 1st and 2nd kinds from

- 24 Dec 1999: rectangular coordinates, loop orientation parameters [PAS]
- 07 Jan 2000: put elliptic integrals inline instead of function calls [PAS]
- 12 Dec 2001: use dipole field as limit at large distances [PAS]
- 14 Dec 2001: change treatment of points very close to wire [PAS]
- 11 Mar 2002: protect against singularity at center of point dipole [PAS]

Variables in calling sequence
- X,Y,Z = coordinates where the induction is to be calculated (m) (input)
- PARAMS contains the following 8 parameters:
  - X0,Y0,Z0 = coordinates of the center of the loop (m) (input)
  - alf,bet,gam = direction cosines of loop axis orientation (input)
  - RADIUS = loop radius (m) (input)
  - CURRENT = clockwise about the (alf,bet,gam) direction (A),
    or if RADIUS=0, magnetic dipole moment/\pi (A-m^2) (input)
  - B = magnetic induction vector in lab coordinate system (T) (output)

IMPLICIT NONE
INCLUDE 'constant.inc'
REAL*8   X, Y, Z, PARAMS(8), B(3)

---

SUBROUTINE BREGION(PART, GEOM, ELEMENT, IREG, JSURF, EBDIST)

Determine if PART is in any magnetic field regions as defined by types of
class 60 found in the ELEMENT structure. Returns region number as IREG,
or 0 if not in a field. In overlapping regions, an ad hoc type 69
region is created. Computes distance EBDIST to the nearest magnetic
region boundary in the forward direction, using surface and region
definitions in GEOM. On return, JSURF is the exit surface or zero if
no intersection is found, in which case EBDIST will be a large positive
number. PART is not changed.

P.A.Seeger, 12 Dec 1999. Modeled after TESTIN, DIST, and DTOEX.
03 Oct 2000: test for "on surface" was too sensitive [PAS]
11 Oct 2000: don't look for PARAM(I) if I=0 [PAS]
20 May 2004: enter with IREG=JSURF=-1 to reinitialize [PAS]
09 Jul 2006: MAXM in mc_geom.inc; double # of surfaces; test max;
superpose overlapping regions as a type 69 [PAS]

Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'
INCLUDE 'mc_elmnt.inc'
INCLUDE 'constant.inc'

Variables in calling sequence:
PART = record with particle coordinates (input)
GEOM = structure with definitions of all surfaces and regions (input)
ELEMENT = structure with definitions of all region parameters (input)
IREG = magnetic region particle is in, or 0 if none (output)
JSURF = surface the direction in PART is pointed toward (output)
EBDIST = distance to surface JSURF (m) (output)

RECORD /PARTICLE/ PART
RECORD /MC_GEOM/ GEOM
RECORD /MC_ELEMENT/ ELEMENT
INTEGER*4 IREG, JSURF
REAL*8 EBDIST

No Externals

**********   B W I R E S   **********

SUBROUTINE BWIRES(X, Y, Z, PARAMS, B)

Magnetic induction produced at point (X,Y,Z) by a series of wire segments
with parameters given in PARAMS. The segments are not necessarily co-
planar, and may or may not be closed. The induction is returned in the
3-vector B.

P.A.Seeger, 23 May 2003.

Variables in calling sequence
X,Y,Z = coordinates where the induction is to be calculated (m) (input)
PARAMS (input) contains the following parameters:
   NODES = number of wires + 1
   CURRENT = current in direction of increasing node number (A)
   Xi,Yi,Zi = coordinates of the ends of each wire segment (m)
   B = magnetic induction vector in lab coordinate system (T) (output)
IMPLICIT NONE
INCLUDE 'mc_elmnt.inc'
REAL*8 X, Y, Z, PARAMS(0:*), B(3)
**CAPILLARY**

LOGICAL FUNCTION CAPILLARY(PART, b, K, COSCRIT, ROUGH, L, &
DISTANCE, ISEED)

Find distance that particle PART will travel along a capillary tube of
radius b with constant curvature K before it strikes the surface at a
grazing angle with sine higher than COSCRIT. The surface may have a
roughness parameter. The function returns .TRUE. if the neutron goes
the full length L, or if it survives the first 100 reflections. The
total DISTANCE traveled is always returned. The entrance of the tube is
centered at the origin, the initial direction of the tube is the Z-axis,
and the curvature is horizontal toward the + X-axis. Gravity is neglected
(turned off), so the routine could also be used for X-rays. The position
of the particle is unchanged.

From subroutine DISTORUS, P. A. Seeger, August 20, 2003.

Definitions of STRUCTUREs:

IMPLICIT NONE
INCLUDE 'mc_geom.inc'
INCLUDE 'mc_elmnt.inc'
INCLUDE 'constant.inc'

Variables in calling sequence

- PART = record of particle, in coordinates of tube (input)
- b = cross-section (small) radius of the torus (m) (input)
- K = curvature (1/A) of toroidal tube (m^-1) (input)
- COSCRIT = cosine of the critical angle (input)
- ROUGH = surface roughness parameter (input)
- L = length of region containing capillary tube (m) (input)
- DISTANCE = actual length traveled (m) (output)
- ISEED = random number seed, may be used by ANGTORUS or WOBBLE (in/out)

RECORD /PARTICLE/ PART
REAL*8    b, K, COSCRIT, ROUGH, L, DISTANCE
INTEGER*4 ISEED

Externals
ANGTORUS  DIST      MOVEX     RFLN      WOBBLE
REAL*8    DIST

**CMPLT_PI**

REAL*8 FUNCTION CMPLT_PI(n, k, flag, CMPLT_K, CMPLT_E)

Computation of the complete elliptic integral of the third kind. Also
returns complete integrals of 1st & 2nd kinds in the calling sequence as
CMPLT_K and CMPLT_E. The ranges of the arguments are |n|<1 and 0<=k<1;
if out of range, flag will be .FALSE. and the function will be 0 on
return, though K and E might be correct.

1st: K(k) = Integral (0, pi/2) of 1/sqrt(1 - (k sin(theta))^2) d(theta)
2nd: E(k) = Integral (0, pi/2) of sqrt(1 - (k sin(theta))^2) d(theta)
3rd: PI(n,k) = Integral (0, pi/2) of

-----------------------------------------------
[1 + n \sin(\theta)^2] \sqrt{1 - (k \sin(\theta))^2}


L.L. Daemen, P.A. Seeger [February 13, 2000]
28 Mar 2000: include output of 1st & 2nd kinds in calling sequence [PAS]
12 Dec 2001: total revision: redefined PI, different algorithm [PAS]
11 Jan 2002: additional term in expansion of K and E for k~1 or k~0 [PAS]

Variables in calling sequence:
k = the modulus (0 <= k < 1) [input]
\( n = \) the parameter (-1 < n < 1) for integrals of 3rd kind [input]
flag = .TRUE. for normal return and .FALSE. if |k| > 1 or |n| > 1 [output]
CMPLT_K = complete elliptic integral of 1st kind at k [output]
CMPLT_E = complete elliptic integral of 2nd kind at k [output]
imPLICIT NONE
REAL*8 k, n, CMPLT_K, CMPLT_E
LOGICAL flag

REAL*8 FUNCTION DET_2D(PART, PARAMS, IX, IY, ISEED)

Encode 2D position sensitive detector (type 43). Returns the detected weight, or zero if outside bounds of detector. Encoding mesh can be plane rectilinear or polar, cylindrical, or spherical.
The corresponding "natural" surface are planes perpendicular to the x-, y-, or z-axis (for rectilinear and polar coordinates), cylinders with axis pointing in the x-, y-, or z-direction, and spherical surfaces with the polar axis pointing in the x-, y-, or z-direction.

L.L. Daemen, July 7, 1997
28 Jul 1998: test azimuthal angle modulo 2pi; index PARAMS from 0 [PAS]
05 Aug 1998: allow inverted limits [PAS]
25 Aug 1998: changed azimuthal angle definitions; restructured [PAS]
14 Sep 1998: all REAL*4 changed to REAL*8 (except PLNORM) [PAS]
03 Oct 2000: allow 0.4 digit to be turned on (polarization) [PAS]

IMPLICIT NONE
INCLUDE 'mc_geom.inc'
INCLUDE 'mc_elmnt.inc'
INCLUDE 'constant.inc'

Variables in calling sequence:
PART = record with detected particle coordinates (input)
PARAMS = array containing the parameters associated with the detector (input)
IX, IY = bin coordinates of detected neutron (output)
ISEED = random number generator seed (input/output)

INTEGER ISEED, IX, IY
REAL*8 PARAMS(0:*)
REAL*4 PLNORM
RECORD /PARTICLE/PART
REAL*8 FUNCTION DIAMOND(LAMBDA, TIN, ABSORB, TWOSINTH, ISEED)

Neutron attenuation length (mm) of polycrystalline diamond sintered in amorphous Cobalt (80 vol% C, 20 vol% Co) at temperature TIN (default 300K), as a function of neutron wavelength LAMBDA in Angstroms. May also return the fraction of the cross section which is nuclear absorption (ABSORB). May also be used to scatter from a random selected plane spacing and return TWOSINTH. A special case is TWOSINTH=2 (or -2), indicating isotropy in C.M. (or lab) system (respectively)

From J. M. Carpenter et al., SIGAL, January 16, 1986 (IPNS Note 32)

18 May 2009: copied from ATTEN_Al with different parameters for diamond, treat Co as amorphous [PAS]
25 May 2009: change normalization to match SIGscat at hi-E; make Co scatter fraction isotropic [PAS]
24 Jun 2009: make test of same LAMBDA single-precision [PAS]

Variables in calling sequence:
LAMBDA = neutron wavelength (A) (input)
TIN = material temperature (K) (optional input)
ABSORB = ratio of absorption/total cross section (optional output)
TWOSINTH = twice the sine of Bragg angle (optional output)
ISEED = random number seed (optional in/out)

IMPLICIT NONE
INCLUDE 'constant.inc'
REAL*8 LAMBDA
REAL*8,OPTIONAL :: TIN, ABSORB, TWOSINTH
INTEGER*4,OPTIONAL :: ISEED

C Externals:
C HUNT8 RAN
REAL*4 RAN
C--
C++
C********** D I G I T S 4 **********
C
SUBROUTINE DIGITS4(Q, STRING, N)

Format a floating-point number into 10 or fewer ASCII characters.
This subroutine encodes a floating-point number Q into a character string no longer than STRING (maximum 10), with no embedded blanks or trailing insignificant zeros. The actual length is returned as N. Underflows are returned as '.000' and overflows as '*****'. Zero Q becomes '0'.

P. A. Seeger, Los Alamos National Laboratory, May 24, 1986
05 Dec 1987: improved elimination of trailing insignificant digits [PAS]
19 Sep 1998: changed name from DIGITS [PAS]
C
C No Externals
C--
C++
C**********  D I G I T S  8 **********
C
SUBROUTINE DIGITS8(Q, STRING, N)
C
C Format a floating-point number into 14 or fewer ASCII characters.
C This subroutine encodes a floating-point number Q into a character string
C no longer than STRING (maximum 14), with no embedded blanks or trailing
C insignificant zeros. The actual length is returned as N. Underflows are
C returned as '.0000' and overflows as '*****'. Zero Q becomes '0'.
C
C P. A. Seeger, Los Alamos National Laboratory, May 24, 1986
C 05 Dec 1987: improved elimination of trailing insignificant digits [PAS]
C 14 Sep 1998: new name; all REAL*4 changed to REAL*8; 14 digits [PAS]
C 11 Jan 1999: 2 more places to right of decimal [PAS]
C
C No Externals
C--
C++
C**********  D I S T  **********
C
REAL*8 FUNCTION DIST(PART, SURF, ON_SURF, DREF)
C
C Find distance along trajectory of particle PART to surface SURF, relative to
C distance DREF (note that DREF is double precision). For particles with
C mass, gravity is included in determining intersection, but the distance
C returned assumes constant velocity. If logical ON_SURF is .TRUE., the
C particle is assumed to be on the surface already; then the computed
C distance to the surface is returned in DREF and the test is made for a
C second intersection with quadratic surface. If the trajectory does not
C intersect the surface, DIST will be -10**38 on return. No parameters of
C the particle are changed.
C
C M.W. Johnson and C. Stephanou, Rutherford Laboratory report RL-78-090.
C Modified by P.A. Seeger (1980) to include test for being on surface ISURF.
C 08 Feb 1994: converted from COMMON to STRUCTUREs; changed calling
C sequence; added gravity [PAS]
C 11 Apr 1994: changed test of AA << BB to avoid underflow [PAS]
C 06 Mar 1995: solve in double precision [PAS]
C 23 Mar 1995: expand square root if BB**2 >> CC/AA [MFitz,PAS]
C 09 Apr 1996: local copies of (X,Y,Z) and (DX,DY,DZ) [PAS]
C 08 Sep 1998: DREF in calling sequence and algorithm [PAS]
C 14 Sep 1998: change function from REAL*4 to REAL*8 [PAS]
C 18 Aug 2006: use DREF only when ON_SURF is true [PAS]
C 11 Sep 2006: don't even set DREF=0 if ON_SURF is false [PAS]
C
C Definitions of STRUCTUREs:
C IMPLICIT NONE
C INCLUDE 'mc_geom.inc'
C INCLUDE 'constant.inc'
C
C Variables in calling sequence
C PART = record with particle coordinates (input)
C SURF = record with surface definition (input)
C I = surface to which distance is being found (input)
C ON_SURF = flag that particle is on the surface in question (input)
C DREF = computed distance to reference surface (m) (input/output)
RECORD /PARTICLE/ PART
RECORD /SURFACE/ SURF
LOGICAL ON_SURF
REAL*8 DREF
C
C  No externals.
C--
C++
C**********  D I S T O R U S  **********
C
REAL*8 FUNCTION DISTORUS(PART, PARAM, ON_SURF, INSIDE)
C
Find distance along trajectory of particle PART to a toroidal surface
defined in PARAM. Only the outer half-shell of the near half of the
torus is accepted. Gravity is included. The plane of the torus may
be horizontal or vertical. If logical ON_SURF is .TRUE., particle is
assumed to be on the torus already, but test is made for an additional
intersection. Flag INSIDE will be set to show if the particle is in the
interior of the torus. If the trajectory does not intersect the torus,
DISTORUS will be -10**38. No parameters of the particle are changed.
C
C 03 Aug 1996: heavily adapted into form of subroutine DIST; omit solutions
on far side of torus; omit inner half-shell; bracket the
root of interest and use inline bisection/Newton-Raphson;
series expansions to improve precision of sqrt near 1
[PAS,MFitz,TLK]
C 08 Aug 1996: add gravity; add INSIDE to calling sequence [PAS]
C 28 Aug 1996: fix error in velocity component VX when vertical [PAS]
C 23 Nov 1996: restructured bracketing of root [PAS]
C 06 Dec 1996: completely new method to find quartic equation from points
   instead of derivatives [PAS]
C 08 Dec 1996: error (since 23 Nov) factor /VXZ in T1-T4 [JRDC,PAS]
C 14 Sep 1998: change function from REAL*4 to REAL*8 [PAS]
C
Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'
INCLUDE 'mc_elmnt.inc'
INCLUDE 'constant.inc'
C
Variables in calling sequence
C PART = record with particle coordinates (input)
C PARAM = block of data with definition of the torus (input)
C ON_SURF = flag that particle is on the surface in question (input)
C INSIDE = flag that particle is inside the volume of the torus (output)
RECORD /PARTICLE/ PART
REAL*8 PARAM(0:*)
LOGICAL ON_SURF, INSIDE
C
C No externals
C--
C++
C**********  D I S T _ A  **********
C
REAL*8 FUNCTION DIST_A(PART, SURF, ON_SURF, DREF, ACCEL)
C
Find distance along trajectory of particle PART to surface SURF, relative
to distance DREF (note that DREF is double precision). The (small)
acceleration ACCEL is included in determining intersection, but the
distance returned assumes constant velocity. If logical ON_SURF is
.TRUE., the particle is assumed to be on the surface already; then the
computed distance to the surface is returned in DREF and the test is made
for a second intersection with quadratic surface. If the trajectory does
not intersect the surface, DIST will be \(-10^{38}\) on return. No parameters
of the particle are changed.

P.A. Seeger, 20 Sep 2003, copied from DIST.

Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'
INCLUDE 'constant.inc'

Variables in calling sequence
PART = record with particle coordinates (input/output)
SURF = record with surface definition (input)
I = surface to which distance is being found (input)
ON_SURF = flag that particle is on the surface in question (input)
DREF = computed distance to reference surface (m) (input/output)
ACCEL = constant acceleration vector (m/us^2) (input)
RECORD /PARTICLE/ PART
RECORD /SURFACE/ SURF
LOGICAL ON_SURF
REAL*8 DREF, ACCEL(3)

No externals.

SUBROUTINE DTOEX(PART, GEOM, IREG, ISURF, JSURF, EXDIST)

Compute distance EXDIST to nearest boundary of region IREG from the
position and velocity given in PART, using surface and region
definitions in GEOM. If IREG = 0, finds distance to nearest
surface in forward direction. If ISURF is not 0, the particle
is assumed to be on that surface already, but may intersect it again
if the surface is quadratic. On return, JSURF is the exit surface
and EXDIST is the distance to it; the particle has not been moved
unless it was necessary to "back up" to surface ISURF. If no exit
is found, JSURF will be 0 and EXDIST will be negative.

M.W. Johnson and C. Stephanou, Rutherford Laboratory report RL-78-090.
Modified by P.A. Seeger (1980) for case of particle on surface ISURF.
02 Jan 1985: Included JSURF as separate entry in calling sequence [PAS]
03 Feb 1994: converted from COMMON to STRUCTUREs; added PART and GEOM
to calling sequence [PAS]
06 Mar 1995: allow zero as a result [PAS]
08 Sep 1998: new calling sequence for DIST, correct for computed
distance to surface ISURF; may back up to ISURF [PAS]
24 Sep 1998: change all REAL*4 to REAL*8; don't back up if JSURF=0 [PAS]
26 Jul 1999: test range of ISURF [PAS]
18 Aug 2006: only apply DREF when ON_SURF is true [PAS]

Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'

Variables in calling sequence:
PART = record with particle coordinates (input/output)
GEOM = structure with definitions of all surfaces and regions (input)
IREG = region particle is in (input)
ISURF = surface which the particle PART is on (input)
JSURF = surface the direction in PART is pointed toward (output)
EXDIST = distance to surface JSURF (output)

record /PARTICLE/ PART
record /MC_GEOM/ GEOM
integer*4 IREG, ISURF, JSURF
real*8 EXDIST

externals:
DIST MOVEX
real*8 DIST

*** E L L I P T I C  ***

SUBROUTINE ELLIPTIC(phi, k, c_ell_k, c_ell_e, i_ell_f, i_ell_e, &
& zeta)

Computation of the complete and incomplete elliptic integrals of the
first and second kind by the method of Landen transformations.
Jacobi's Zeta function is also returned. The required precision eps is
specified in a parameter statement

Reference: L.V. King, "On the direct numerical calculation of elliptic
functions and integrals," (Cambridge University Press,
Cambridge, 1924).

L.L. Daemen and P.A. Seeger [February 13, 2000]
01 Apr 2000: protect against k^2>1 and phi=n(pi/2) [PAS]
03 Apr 2000: fixed power of Z; use atan instead of atan2 [LLD,PAS]
21 Oct 2002: change abs(mod(phi0... to dabs(dmod(phi0... [PAS]

implicit none
include 'constant.inc'
real*8 eps
parameter (eps=1.0d-5)

Variables in calling sequence:
phi = the argument of the (incomplete) elliptic functions [K(k)=F(pi/2,k),
and E(k)=E(pi/2,k)], (0 < phi < pi/2) [input]
k = the modulus (k=sin(alpha) in Abramowitz and Stegun notations) [input]
c_ell_k = the complete elliptic integral of the first kind at k [output]
c_ell_e = the complete elliptic integral of the second kind at k [output]
i_ell_f = the incomplete elliptic integral of the first kind at
phi and k [output]
i_ell_e = the incomplete elliptic integral of the second kind at
phi and k [output]
zeta = Jacobi's zeta function evaluated at phi and k [output]

real*8 phi, k, i_ell_f, i_ell_e, c_ell_e, c_ell_k, zeta

No externals

*** E L S C A T  ***

SUBROUTINE ELSCAT(VX, VY, VZ, STH, ISEED)

Elastic scattering from original REAL*4 velocity (VX,VY,VZ) at random
azimuthal angle about the velocity vector, at a scattering angle
given by STH = 2 sin(theta/2). If the velocity is 0 or if the
Variables in calling sequence:
VX, VY, VZ = vector velocity (input/output)
STH = twice the sine of half the scattering angle (input)
ISEED = random-number generator seed (input/output)

Implicit
NONE
REAL*4 VX, VY, VZ
REAL*8 STH

Externals
RNDCRCL

********** ELSCAT2 **********

SUBROUTINE ELSCAT2(VX, VY, VZ, STH, PHI1, PHI2, FLAG, ISEED)

Elastic scattering from original REAL*4 velocity (VX, VY, VZ) at a
scattering angle given by \( STH = 2 \sin(\theta/2) \), with the azimuthal
compontent of the scattering vector restricted to the angular range
[PHI1,PHI2], measured from the horizontal plane. If the initial
c velocity is too vertical, or the absolute value of STH>2, then no
scattering occurs and FLAG is set to .FALSE..

L.L.Daemen & P.A.Seeger, 14 May 1998; derived from ELSCAT.
14 Sep 1998: change all REAL*4 to REAL*8 (except RAN) [PAS]
28 Sep 1998: change VX, VY, VZ back to REAL*4 [PAS]

Variables in calling sequence:
VX, VY, VZ = vector velocity (input/output)
STH = twice the sine of half the scattering angle (input)
PHI1,PHI2 = limits of azimuthal angle (input)
FLAG = flag set to .FALSE. if no scattering (output)
ISEED = random-number generator seed (input/output)

Implicit
NONE
REAL*4 VX, VY, VZ
REAL*8 STH, PHI1, PHI2
LOGICAL FLAG
INTEGER*4 ISEED

Externals
RAN

********** ERF **********

REAL*8 FUNCTION erfc(X)

Function to compute complementary error function to a fractional
precision ~ 1.2e-7. The cumulative normal distribution to z is
erfc[-z/sqrt(2)]/2.
IMPLICIT NONE
REAL*8   X

No Externals
C--
C********** F E R M I **********
C
LOGICAL FUNCTION FERMI (PART, PARAM, EXDIST, ISEED)

Subroutine to simulate a Fermi chopper rotating about the vertical (Y) axis. The chopper has a cylindrical body with a central rectangular hole for a slit package. Blades may be curved or straight. The neutron trajectory is approximated by a quadratic with respect to any surface, assuming the transit time is short compared to the rotation period. For subtype 0, the body is transparent and the blades are perfect absorbers. For subtype 1, body and blade attenuations are computed as non-resonant materials such as Al or B4C, while for subtype 2 the blades are a resonant absorber such as Cd or Gd. The spaces between blades are always transparent. The function is .FALSE. if a neutron penetrates any material or gets through when the rotor phase is backwards, and .TRUE. if neutron passes forward without attenuation or is totally absorbed.

L.L. Daemen - February 1998
P.A. Seeger - 26 July 1998
14 Sep 1998: change all REAL*4 to REAL*8 (except PLNORM) [PAS]
14 Aug 2002: *MAJOR REVISION*; add parameters (length & spacing of slits);
             more precise attenuation; resonant absorbers for blades [PAS]
21 Oct 2002: change MOD() to AMOD() for F90 standard [PAS]

Definitions of STRUCTURES:
IMPLICIT NONE
INCLUDE 'mc_elmnt.inc'
INCLUDE 'mc_geom.inc'
INCLUDE 'constant.inc'

Variables in calling sequence
PART = record with particle coordinates (input/output)
PARAM = block of data with definition of the Fermi chopper (input)
EXDIST = distance to exit surface particle is aimed at (m) (input/output)
ISEED = random-number generator seed (input/output)
REAL*8   PARAM(0:*), EXDIST
INTEGER*4 ISEED
RECORD /PARTICLE/ PART

Externals
PLNORM    SINGLEV
REAL*4    PLNORM
REAL*8    SINGLEV
C--
C**********   G A M M L N   **********
C
REAL*8 FUNCTION GAMMLN(X)
C
Returns natural log of gamma(X), for X > 0.
SUBROUTINE GET_BINS(PARAMS, IXMAX, IYMAX, ITMAX, &
                    DELX, DELY, T_BINS, XMIN, YMIN)

Look up the binning parameters for the detector given in structure
PARAMS, and return dimensions, pixel sizes, the time-slice boundaries
(note that number of boundaries is 1 more than number of bins), and
minimum horizontal and vertical detector edges.

P. A. Seeger, April 10, 1994
04 Jan 1995: MC_ELMNT.INC now has parameter offsets, not UNIONs [PAS]
28 Mar 1995: new calling sequence to return XMIN and YMIN; added type 44;
default DELX and DELY = 0 (was 1 m) [PAS]
12 Apr 1995: allow linear t bins; test for t valid scale; offset t [PAS]
26 Jul 1995: linear scale always started at zero [PAS]
24 Jan 1996: change sign of D_DELAY; compute T in double precision [PAS]
18 Sep 1997: allow D_TICK=0 for perfect log scale [PAS]
14 Sep 1998: change REAL*4 to REAL*8, except T_BINS [PAS]
16 Sep 2009: use |DET_NX| and |DET_NY| (negative for array of tubes) [PAS]

Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'
INCLUDE 'mc_elmnt.inc'

Variables in calling sequence:
PARAMS = array with description of a detector (input)
IXMAX = number of horizontal detector elements (output)
IYMAX = number of vertical detector elements (output)
ITMAX = number of time bins (output)
DELX, DELY = size of detector elements (m) (output)
T_BINS = array of ITMAX+1 time bin boundaries (us) (output)
XMIN, YMIN = detector boundaries, horizontal and vertical (m) (output)
INTEGER IXMAX, IYMAX, ITMAX
REAL*4 T_BINS(*)
REAL*8 PARAMS(0:*), DELX, DELY, XMIN, YMIN

No Externals
--
++

SUBROUTINE GET_BMAP(PART, DIST, GEOM, ELEMENT, NS, t, X, Y, Z, &
                     Bx, By, Bz)
Map the magnetic induction field vector \((B_x, B_y, B_z)\) (T) along the trajectory of particle PART at NS equal steps over the distance DIST. The time \(t\) (us) and position \((X, Y, Z)\) (m) are also returned in arrays of length NS. If DIST or the velocity is 0, only the initial point will be calculated and NS will be set to 1. The trajectory includes gravity unless PART.M is 0. PART is not changed.


01 Apr 2000: new calling sequence for BFIELD [PAS]

Definitions of STRUCTUREs:

```
IMPLICIT NONE
INCLUDE 'mc_geom.inc'
INCLUDE 'constant.inc'
```

Variables in calling sequence:

- PART = record containing particle coordinates & velocity (input)
- DIST = distance to track trajectory of particle (input)
- GEOM = structure with all surface and region definitions (input)
- ELEMENT = structure with region space and magnetic parameters (input)
- NS = number of steps into which to divide DIST (input/output)
- t = time-of-flight at each step (us) (output)
- X, Y, Z = particle position at each step (m) (output)
- Bx, By, Bz = magnetic induction vector at each step (T) (output)

```
RECORD /PARTICLE/ PART
RECORD /MC_GEOM/ GEOM
RECORD /MC_ELEMENT/ ELEMENT
INTEGER NS
REAL*8  DIST, t(*), X(*), Y(*), Z(*), Bx(*), By(*), Bz(*)
```

Externals:

```
BFIELD  BREGION
```

C**********  G E T _ R H O  **********

```
COMPLEX FUNCTION GET_RHO(LAMBDA, PARAMS, RHOM)
```

Look up the complex scattering-length density \((10^{10}/cm^2)\) for a region with type and parameters given in structure PARAMS. This may be a function of the neutron wavelength, LAMBDA. For magnetic materials, the complex magnetic scattering-length density is returned in RHOM.

P. A. Seeger, March 28, 1988

17 Apr 1994: include supermirror type (4); ISEED in calling sequence [PAS]
05 Jan 1995: MC_ELMNT.INC now has parameter offsets, not UNIONs [PAS]
16 Oct 1995: always return normal RHO for supermirror; remove ISEED [PAS]
14 Sep 1998: change function to COMPLEX*16, PARAMS to REAL*8 [PAS]
06 Mar 1999: revert to COMPLEX*8 to match F90 standard [PAS]
18 Sep 1999: add type 7; added LAMBDA to calling sequence [PAS]
28 Sep 2000: allow subtypes; added RHOM to calling sequence [PAS]
19 Aug 2002: new calling sequence for SINGLEV with polarization; change RHOM to COMPLEX [PAS]
04 Aug 2006: explicit values for Al (type 2) and Be (type 5) [PAS]

```
IMPLICIT NONE
INCLUDE 'mc_geom.inc'
INCLUDE 'mc_elmnt.inc'
```

Variables in calling sequence:
B-20  MCLIB

LAMBDA = neutron wavelength (A) (input)
PARAMS = parameters describing material in region (input)
RHOM = magnetic scattering-length density (10^-10/cm^2) (output)

REAL*8  LAMBDA, PARAMS(0:*)
COMPLEX*8  RHOM

C Externals:
SINGLEV

C***********    G I N I    **********

LOGICAL FUNCTION GINI(X, Y, YVAR, NCHANS, INTENS, SIGINT, CENTER, 
&
SIGCEN, SIGMA)

This routine sums data and computes the centroid and estimates the 
standard deviation of the distribution using Gini's mean-difference 
integrated within and summed over histogram bins. The value .TRUE. 
is returned if successful, .FALSE. if too many negative data in the 
array.

P. A. Seeger, Los Alamos National Laboratory, Feb. 5, 1992
14 Sep 1998: change all REAL*4 to REAL*8 [PAS]
22 Aug 2004: revised algebra of Gini sums, results unchanged [PAS]

Variables in calling sequence
X       R(*)    Input     Array of bin boundaries (NCHANS + 1)
Y       R(*)    Input     Array of counts-per-bin
YVAR    R(*)    Input     Array of variances of Y; if Y is Poisson, 
this may be the same array as Y
NCHANS  I       Input     Length of arrays; number of bins to sum
INTENS  R       Output    Integrated intensity (sum of counts)
SIGINT  R       Output    Standard deviation of INTENS
CENTER  R       Output    Centroid of distribution
SIGCEN  R       Output    Standard deviation of CENTER
SIGMA   R       Output    Gini's mean-difference estimator of std. dev.

C**********  G R A V _ F O C  **********

REAL*8 FUNCTION GRAV_FOC(T, ACCEL, JITTER, START, TOP, PERIOD, 
&
ISEED)

Determine vertical position of gravity focusing device.
P. A. Seeger, April 7, 1994
14 Sep 1998: change all REAL*4 to REAL*8 (except PLNORM) [PAS]
20 May 2004: enter with ISEED=0 to reinitialize [PAS]

Variables in calling sequence:
T = time (measured from proton pulse) at which to determine position (us)
ACCEL = acceleration of gravity focuser (m/us/us)
JITTER = rms phase uncertainty of gravity focuser (us)
START = earliest time that gravity focuser must accept (us)
TOP = latest time, top of trajectory (us)
PERIOD = time between successive pulses of the gravity focuser (us)
ISEED = random number generator seed
REAL*8  T, ACCEL, JITTER, START, TOP, PERIOD
INTEGER ISEED
C C Externals:
C PLNORM
    REAL*4 PLNORM
C--
C++
C********** H U N T 4 **********
C
SUBROUTINE HUNT4(XX, N, X, JLO)
C
Find index JLO in array XX of length N, such that X is between XX(JLO)
and XX(JLO+1). If JLO = 0 at entry, use full bisection; otherwise
begin search at initial value of JLO.
C
Press et al., "Numerical Recipes" (Cambridge, 1986) p. 91;
adapted by P. A. Seeger, April 13, 1987.
C
No externals
C Variables in calling sequence:
C XX     R(*)    Input     Monotonic array
C N      I       Input     Length of array XX
C X      R       Input     Value of which to find location in XX
C JLO    I       In/Out    Index in XX such that XX(JLO) and XX(JLO+1)
    bracket X
C--
C++
C********** H U N T 8 **********
C
SUBROUTINE HUNT8(XX, N, X, JLO)
C
Find index JLO in array XX of length N, such that X is between XX(JLO)
and XX(JLO+1). If JLO = 0 at entry, use full bisection; otherwise
begin search at initial value of JLO.
C
Press et al., "Numerical Recipes" (Cambridge, 1986) p. 91;
adapted by P. A. Seeger, April 13, 1987.
C 14 Sep 1998: new name, from HUNT, change REAL*4 to REAL*8 [PAS]
C
No externals
C Variables in calling sequence:
C XX     R(*)    Input     Monotonic array
C N      I       Input     Length of array XX
C X      R       Input     Value of which to find location in XX
C JLO    I       In/Out    Index in XX such that XX(JLO) and XX(JLO+1)
    bracket X
C--
C++
C********** I M P U L S E **********
C
SUBROUTINE IMPULSE(PARAMS, E0, E1, COS_TH, PDCS, ISEED, IFLAG)
c
Subroutine to scatter neutrons according to the impulse approximation
c of deep inelastic neutron scattering (DINS). Atoms are treated as non-
c interacting, which requires Q>>2pi/(interatomic distance). y-scaling
c is used, where y = longitudinal momentum = M(E0-E1)/hbar^2Q - Q/2, and
c y is chosen from a normal distribution. Q is chosen from a quadratic
c distribution with (Qmin,Qmax) limited by dynamics. Output values of E1
c and COS_TH are derived from Q and y. The double differential cross
section normalization PDCS is computed, including terms representing an
expansion of the Compton profile and first-order final-state effects.
IFLAG is a diagnostic, the number of tries to get a valid (y,Q). If
no valid (y,Q) is found in 10 tries, then IFLAG=atom index, E1=E0,
COS_TH=1, and PDCS=0.

Derived from codes PDCS and TEST_PDCS by Roberto Senesi
P.A. Seeger, 24 Nov 2007

08 Apr 2008: increase sampling rate of tails of y [PAS]
10 Apr 2008: limit profile correction [PAS]
11 Apr 2008: special case for sig(y)=0 [PAS]
26 Apr 2008: restructured and renormalized [PAS]
28 May 2008: analytic forms for ymax, Qmax, Qmin [PAS]

C Include files with parameter list definitions and constants
IMPLICIT  NONE
INCLUDE 'mc_elmnt.inc'
INCLUDE 'constant.inc'

C Variables in calling sequence:
PARAMS = block of region parameters (input)
E0 = energy of the incident neutron in meV (input)
E1 = energy of the scattered neutron in meV (output)
COS_TH = cosine of the scattering polar angle (output)
PDCS = partial differential cross section normalization (output)
ISEED = seed for the random number generator (input/output)
IFLAG = + number of tries to select (y,Q), or - atom index (output)
REAL*8 PARAMS(0:*), E0, E1, COS_TH, PDCS
INTEGER*4 ISEED, IFLAG

C Externals:
erfc      PLEXP     PLNORM    RAN0
REAL*8  erfc
REAL*4  PLEXP,PLNORM,RAN0

C**       I N T E R P 4 **********
C
SUBROUTINE INTERP4(XX, X, NX, I, R)
C
Subroutine to interpolate a value XX in a table X of length NX. Result
is index number I of the cell containing XX, and the ratio R of XX
into the cell. Input array X must be monotonic increasing, with no
repeated values. If XX is outside the range of the table, I will
be 1 and R negative or I = NX-1 and R > 1, indicating extrapolation.

P. A. Seeger, 1984
23 May 2006: correct upper index for extrapolation [PAS]

C Variables in calling sequence:
XX = value to be found in table
X = array in which to interpolate
NX = length of array X
I = index of table value next lower (or =) XX
R = ratio of location of XX to X(I+1) - X(I)
IMPLICIT  NONE
INTEGER*4 NX, I
REAL*4   XX, X(NX), R
SUBROUTINE INTERP8(XX, X, NX, I, R)

C Subroutine to interpolate a value XX in a table X of length NX. Result
C is index number I of the cell containing XX, and the ratio R of XX
C into the cell. Input array X must be monotonic increasing, with no
C repeated values. If XX is outside the range of the table, I will
C be 1 and R negative or I = NX-1 and R > 1, indicating extrapolation.

C P. A. Seeger, 1984
C 14 Sep 1998: new name, from INTERP, change REAL*4 to REAL*8 [PAS]
C 23 May 2006: correct upper index for extrapolation [PAS]

C Variables in calling sequence:
C XX = value to be found in table
C X = array in which to interpolate
C NX = length of array X
C I = index of table value next lower (or =) XX
C R = ratio of location of XX to X(I+1) - X(I)

IMPLICIT NONE
INTEGER*4 NX, I
REAL*8 XX, X(NX), R

C No externals
C--
C++

SUBROUTINE KERNEL(FNAME, E, EPRIME, COS2TH, SIGSCAT, ISEED)

C Subroutine to scatter neutrons according to the S(alpha,beta) scattering
C law tabulated in file FNAME, which is a Type 1 thermal data table (NTY=4)
C from the MCNP library. Elastic and inelastic cross sections and the
C sum SIGSCAT are found for incident energy E (or EPRIME) as in MCNP
C subroutine acetot.F90, and final energy EPRIME and direction cosine
C COS2TH are found as in MCNP subroutine sabcol.F90. Only one material
C file at a time is stored in allocated memory.

C L.L. Daemen, P.A. Seeger - April 1, 1995; reference MCNP Manual LA-12625-M,
C issued Nov. 1993, Judith F. Briesmaster, ed.
C 03 Apr 1995: eliminate local arrays, use XSS directly [PAS,LLD]
C 14 Sep 1998: change all REAL*4 to REAL*8 (except RAN); energy
C units meV [PAS]
C 24 Aug 1999: make case of variables consistent, for non-standard Absoft
C compiler [LLD]
C 03 Feb 2003: added SCATFRAC to call, changed names in call [PAS]
C 20 May 2004: enter with ISEED=0 to reinitialize; use unit 34 [PAS]
C 20 Mar 2009: use FNAME as flag for initialization [PAS]
C 26 Mar 2009: eliminate MFP & SCATFRAC, return total cross section
C (SIGSCAT) at energy EPRIME [PAS]
C 30 Mar 2009: end values of inelastic energy have lower probability [PAS]
C 03 Apr 2009: mode 4, exact coherent treatment for Be & graphite, multiply
C elastic xsec by 10^9 meV/MeV, force sigelas=0 at low E [PAS]
C 15 Apr 2009: RAN0 instead of RAN; XSS is allocatable; implement mode 2
C (continuous angle) for both elastic & inelastic [PAS]
IMPLICIT NONE

Variables in calling sequence:
- FNAME = path to the file containing the scattering law (in/out)
- E = energy of the incident neutron in meV (input)
- EPRIME = energy of the scattered neutron in meV (in/out)
- COS2TH = cosine of the scattering angle, 2*theta (output)
- SIGSCAT = total scattering cross section at EPRIME, in b (output)
- ISEED = seed for the random number generator (in/out)

CHARACTER FNAME*40
REAL*8 E, EPRIME, COS2TH, SIGSCAT
INTEGER*4 ISEED

Externals:
- INTERP8, RAN0
REAL*4 RAN0

Logical function LIGHTRFL(COSINC, REFINDX, PROB, ISEED)

Determine probability of reflection of light at a boundary between media with different indices of refraction. Arguments are
- COSINC = the cosine of the angle of incidence, and REFINDX = (index of refraction of new region)/(index of refraction of old region).
- The two polarizations are averaged, and no estimate of the polarization after reflection is made. The function is set to .TRUE. if reflection will occur (based on a random number), and .FALSE. if not. The computed reflection probability PROB is also returned.

P. A. Seeger, 19 Mar. 1985. Modified from LREFLCT.
14 Sep 1998: change all REAL*4 to REAL*8 (except RAN) [PAS]

Variables in calling sequence:
- COSINC = cosine of angle of incidence (input)
- REFINDX = relative index of refraction of next region (input)
- PROB = reflection probability (output)
- ISEED = random-number generator seed

Implicit None
INTEGER*4 ISEED
REAL*8 COSINC, REFINDX, PROB

External:
RAN
REAL*4 RAN

Logical function LMONOCRM(PART, PARAMS, GEOM, IREG, JSURF, & KSURF, EXDIST, NSTEPS, ISEED)

Transport the neutron described in PART through a monochromator crystal described by mosaic spread and ideal reflection probabilities given in PARAMS, and whose region definition is IREG and entrance surface JSURF.
The reflecting planes are not necessarily parallel to JSURF, and the neutron may be reflected multiple times. Each step is computed based on the Darwin equations and the kinematical model (Sears, Acta Cryst. A53 (1997) 35-45). Effects included are Bragg scattering, incoherent and coherent-inelastic scattering, and nuclear absorption. The function will
be true if an odd number of reflections occurs. On return, KSURF will be
the exit surface from the region and EXDIST will be the distance to KSURF.
The number of attempted reflections is returned as NSTEPS, which is set
to negative if the neutron scattered.

P.A. Seeger and Mike Fitzsimmons, June 8, 1995
21 Sep 1995: better choice of IAXIS to minimize SIN/FF(IAXIS) [PAS,GXu]
30 Oct 1997: Lorentzian 2d-spacing if width<0; allow 0 rotation [PAS]
27 Nov 1997: protect against required cos_th>1 [LLD,PAS]
04 Dec 1997: save d-spacing for last if larger effect [PAS]
14 Sep 1998: change all REAL*4 to REAL*8 (except PLNORM, RAN) [PAS]
13 May 1999: added vector TAU for reflecting plane orientation; mosaic
spread has 2 instead of 3 components; fixed error in width
of Lorentzian; use rotation matrix. [PAS]
19 May 1999: ***NEW CALLING SEQUENCE***; do the reflections and move
the particle in multiple steps; absorption & scatter [PAS]
04 Jun 1999: limit backscatter singularity in kinematic model [PAS]
22 Apr 2000: correct sign error in final rotation, normalization [PAS]
17 Feb 2001: change to use full solution of Darwin equations, from
18 Feb 2001: adjust final step to reach surface exactly [PAS]
21 Feb 2001: add Laue geometry test for mosaic blocks [PAS]
03 Jan 2005: accumulate internal motions in REAL*8 [PAS]
16 May 2006: fix sign of SIN error (since 22 Apr 00) when IAXIS=1 [PAS]

Definitions of STRUCTUREs and physical constants:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'
INCLUDE 'mc_elmnt.inc'
INCLUDE 'constant.inc'

Variables in calling sequence:
PART = record containing particle coordinates (input)
PARAMS = array with parameters describing mosaic crystal (input)
GEOM = structure with full description of instrument (input)
IREG = region number in the GEOM structure (input)
JSURF = surface number for entrance of region (input)
KSURF = surface number for exit from region (input/output)
EXDIST = distance to exit of region (m) (input/output)
NSTEPS = number of attempted reflections; - if scattered (output)
ISEED = random-number generator seed (input/output)

RECORD /PARTICLE/ PART
RECORD /MC_GEOM/ GEOM
REAL*8 PARAMS(0:NUMBER_13-1), EXDIST
INTEGER IREG, JSURF, KSURF, NSTEPS
INTEGER*4 ISEED

Externals:
ANGLI DTOEX MOVEX NORM8 PLNORM RAN
REAL*4 PLNORM, RAN

********* L O N G O U T **********
SUBROUTINE LONGOUT(Q, N, IUNIT)

Output an array of N (up to 15000) real*4 numbers Q to unit IUNIT, as ASCII
characters in free format with between 5-1/2 and 7 significant digits,
separated by commas, as a single record terminated with a <blank><CR><LF>.
LOGICAL FUNCTION LORRAND(CXMIN, CXMAX, CYMIN, CYMAX, CZMIN, CZMAX,&
                     VX, VY, VZ, ISEED)

C  Generate random orientation REAL*4 unit vector (VX,VY,VZ), in solid angle
C  specified by direction cosine ranges [CXMIN,CXMAX], [CYMIN,CYMAX], and
C  [CZMIN,CZMAX]. If no solution is possible, the function is .FALSE. and
C  a zero vector is returned.
C
C  04 Dec 1997: Luke Daemen and Phil Seeger
C  14 Sep 1998: change REAL*4 to REAL*8 (except RAN0, velocity) [PAS]
C  30 Dec 2004: account for precision in test for 0 solid angle [PAS]
C
C Variables in calling sequence:
C  CXMIN,CXMAX = minimum and maximum values of the x direction cosine
C  CYMIN,CYMAX = minimum and maximum values of the y direction cosine
C  CZMIN,CZMAX = minimum and maximum values of the z direction cosine
C  VX, VY, VZ = direction cosines, REAL*4 (output)
C  ISEED = random-number generator seed (input/output)
C
IMPLICIT NONE
INTEGER*4 ISEED
REAL*8    CXMIN, CXMAX, CYMIN, CYMAX, CZMIN, CZMAX
REAL*4    VX, VY, VZ

C Externals
C ORRAND    RAN0      RNDCRCL
REAL*4 RAN0

LOGICAL FUNCTION LREFLCT(SINPHI, CXRATIO, PROB, ISEED)

C  Determine probability of reflection of a neutron at a boundary between
C  media with different coherent scattering lengths. Arguments are
C  SINPHI = the sine of the angle from the surface (glancing angle),
C  and CXRATIO = Lambda**2/(2*Pi) times the coherent scattering length
C  ratio (i.e., coherent scattering length per unit volume on the far side
C  of the surface divided by the near side), which is a complex quantity
C  to account for absorption. The function is set to .TRUE. if reflection
C  will occur (based on a random number), and .FALSE. if not. The computed
reflection probability PROB is also returned.


07 Jan 1985: Test for valid SINPHI [PAS]
08 Sep 1997: internal computation in double precision [PAS, LLD]
14 Sep 1998: change all REAL*4 to REAL*8, COMPLEX to COMPLEX*16 [PAS]
08 Mar 1999: revert to COMPLEX*8 to match F90 standard [PAS]

Variables in calling sequence:
SINPHI = sine of glancing angle (cosine of angle of incidence) (input)
CXRATIO = 1 - relative index of refraction of next region (input)
PROB = reflection probability (output)
ISEED = random-number generator seed (input/output)

IMPLICIT NONE
INTEGER*4 ISEED
REAL*8 SINPHI, PROB
COMPLEX*8 CXRATIO

External:
RAN
REAL*4 RAN

SUBROUTINE MOVEA(PART, EXDIST, ACCEL)

Advance the position of particle PART by a distance EXDIST, including
a (small) acceleration vector ACCEL. If the distance is insignificantly
small, a very small step is made in the coordinate with the largest
initial velocity. The position, velocim and time of PART are updated,
and EXDIST is set to 0 on return.

P. A. Seeger, 20 Sep 2003. Copied from MOVEX.

Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'
INCLUDE 'constant.inc'

Variables in calling sequence:
PART = record containing particle coordinates (input/output)
EXDIST = distance to move particle (input/output)
ACCEL = constant acceleration vector (m/us^2) (input)
RECORD /PARTICLE/ PART
REAL*8 EXDIST, ACCEL(3)

No externals

SUBROUTINE MOVEB(PART, EXDIST, GEOM, ELEMENT, NSTEPS, PRECES)

Advance the position of particle PART by a distance less than or equal to
EXDIST, including gravitational droop and spin precession in any magnetic
field regions along the path. The position, time, and polarization of PART
are updated, and EXDIST is reduced by the distance moved. The number of
integration steps NSTEPS and amount of precession PRECES are returned for
diagnostic purposes. A second entry point is provided to set a flag for
writing P on unit 60 after every move in a magnetic field region.

P. A. Seeger, 20 Dec 1999 (constant fields).
28 Dec 1999: possible numerical integration, or external integration [PAS]
27 Jan 2000: added NSTEPS and PRECES to calling sequence [PAS]
02 Feb 2000: no internal integration, but fit initial conditions [PAS]
11 Feb 2000: added region name to calls to BFIELD [PAS]
08 Jul 2006: corrected region name in calls to BFIELD [PAS]
06 Nov 2009: added P to call for BFIELD, to return R*8 polarization;
flag to print out steps [PAS]

Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'
INCLUDE 'constant.inc'

Variables in calling sequence:
PART = record containing particle coordinates (input/output)
EXDIST = distance to move particle (input/output)
GEOM = structure with all surface and region definitions (input)
ELEMENT = structure with region space and magnetic parameters (input)
NSTEPS = number of steps in integration (output)
PRECES = total precession (radian) (output)
FLAG = run-time flag for outputting P on unit 60 (input)

RECORD /PARTICLE/ PART
RECORD /MC_GEOM/ GEOM
RECORD /MC_ELEMENT/ ELEMENT
REAL*8 EXDIST, PRECES
INTEGER NSTEPS
LOGICAL FLAG

Externals:
BFIELD BREGION MOVEX

********** MOVEX **********

SUBROUTINE MOVEX(PART, EXDIST)

Advance the position of particle PART by a distance EXDIST, including
gravitational droop. If the distance is insignificantly small, a very
small step is made in the coordinate with the largest velocity. The
position and time of PART are updated, and EXDIST is set to 0 on return.

P. A. Seeger, 1980.
Modified for time dependence, 1984.
08 Feb 1994: converted from COMMON to STRUCTUREs; added PART to calling
sequence; include gravitational droop [PAS]
19 Apr 1994: make sure particle has actually moved [PAS]
14 Sep 1998: change all REAL*4 to REAL*8 [PAS]
03 Apr 2004: change (X,Y,Z) back to REAL*4 [PAS]

Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'
INCLUDE 'constant.inc'

Variables in calling sequence:
MCLIB  B-29

C    PART = record containing particle coordinates (input/output)
C    EXDIST = distance to move particle (input/output)
    RECORD /PARTICLE/ PART
    REAL*8 EXDIST

C No externals
C--
C++
C********** N E X T R G **********
C
INTEGER*4 FUNCTION NEXTRG(PART, GEOM, IREG, ISURF)

C Find what region the particle defined in PART, presently on surface
C ISURF exiting region IREG, will enter on the other side of the surface.
C Returns current region number (IREG) if not on a surface, and if no valid
C region is found from the definitions in GEOM, returns 0 for ordinary
C surface types or IREG if surface type >= 10.  No parameters are changed.
C
C P. A. Seeger, 1980.
C 03 Feb 1994: converted from COMMON to STRUCTURES; added PART and GEOM
to calling sequence  [PAS]
C 29 Jun 1995: if IGEOM1>10, stay in same region instead of being lost [PAS]
C 11 Dec 1999: change test to IGEOM1>=10 for magnetic-region surfaces [PAS]
C 04 May 2002: surface type >20 means check regions on both sides; keep
looking for "real" region; omit test of regions not bounded
by ISURF  [PAS]
C
C Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'
C
C Variables in calling sequence:
C    PART = record with particle coordinates (input)
C    GEOM = structure with all surface and region definitions (input)
C    IREG = region particle was in before striking surface (input)
C    ISURF = surface upon which particle in /MC_PART/ is located (input)
    RECORD /PARTICLE/ PART
    RECORD /MC_GEOM/ GEOM
    INTEGER*4 IREG, ISURF

C Externals:
C    TESTIN
    LOGICAL TESTIN
C--
C++
C**********  N O P T I C S  **********
C
SUBROUTINE NOPTICS(NEUTRON, GEOM, ELEMENT, SURFTYPE, KSURF, IREG, &
& KREG, FIELDS, SECOND_NT, SREG, ISEED)

C Solve for probability of reflection or refraction (including polarization)
C for NEUTRON, crossing surface KSURF from region IREG to region KREG.
C The surface type number is SURFTYPE, and additional surface and region
C parameters are found in GEOM.  The position, direction, and polarization
C of NEUTRON are updated.  At exit, KREG will be the new region number for
C NEUTRON (IREG if reflected, KREG if transmitted, 0 if absorbed).  If surface
C reflection results in polarization > 0.1%, or if reflection probability is
C in the range 1-99%, or if SURFTYPE=6, then a second history will be returned
C as SECOND_NT in region SREG.
Definitions of STRUCTUREs:

IMPLICIT NONE
INCLUDE 'mc_geom.inc'
INCLUDE 'mc_elmnt.inc'
INCLUDE 'constant.inc'

Variables in calling sequence:

NEUTRON = record containing description of particle (input/output)
GEOM = structure with all surface and region definitions (input)
ELEMENT = structure with all region space and magnetic parameters (input)
SURFTYPE = surface type number: 2=reflective, 3=diffuse, 4=absorbing, 6=splitting (input)
KSURF = surface number that particle is on (input)
IREG = region number on the upstream side of the surface (input);
       if absorbed, will be set to downstream region (output)
KREG = region number on the downstream side of the surface (input);
       region number of NEUTRON (output)
FIELDS = flag to include magnetic fields (input)
SECOND_NT = description of split neutron created by operation (output)
SREG = region number for SECOND_NT, IREG or KREG, or 0 if none (output)
ISEED = random-number generator seed (input/output)

RECORD /PARTICLE/ NEUTRON, SECOND_NT
RECORD /MC_GEOM/ GEOM
RECORD /MC_ELEMENT/ ELEMENT
INTEGER SURFTYPE, KSURF, IREG, KREG, SREG, ISEED
LOGICAL FIELDS

Externals

ANGLI BFIELD BREGION GET_RHO LREFLCT ORRAND RAN
RFLN SNELL TESTIN WOBBLE
COMPLEX*8 GET_RHO
REAL*4 RAN
LOGICAL LREFLCT, TESTIN

********** N O R M 4 **********

SUBROUTINE NORM4(X1, X2, X3)

Normalize three elements (X1,X2,X3) to unit vector magnitude (if not 0).

M.W. Johnson and C. Stephanou, Rutherford Laboratory report RL-78-090.

Variables in calling sequence:

X1,X2,X3 = three components of vector to be normalized (input/output)
C  No externals
C--
C++
C**********  N O R M 8  **********
C
SUBROUTINE NORM8(X1, X2, X3)
C
Normalize three elements (X1,X2,X3) to unit vector magnitude (if not 0).
C
M.W. Johnson and C. Stephanou, Rutherford Laboratory report RL-78-090.
C  19 Sep 1998: new name, from NORM, use REAL*8 instead of REAL*4 [PAS]
C
Variables in calling sequence:
X1,X2,X3 = three components of vector to be normalized (input/output)
IMPLICIT NONE
REAL*8  X1, X2, X3
C
C  No externals
C--
C++

C**********  n o r m _ v e c  **********
!
SUBROUTINE norm_vec(nx, ny, nz, x, y, z)
!
!  Compute normal vector (nx,ny,nz) to vector (x,y,z)
!
!  Copied from McStas (ver. 1.7), author & date not indicated.
!
!  Variables in calling sequence
!  nx,ny,nz = normal vector (output)
!  x,y,z = input vector (input)
!  IMPLICIT NONE
!  REAL*8  nx, ny, nz, x, y, z
!
!  No Externals
!--
C++
C**********   N _ S O U R C E  **********
C**********  G E T _ S P A C E  **********
C
LOGICAL FUNCTION N_SOURCE(NEUTRON, PARAMS, SURF, GFACCEL_2, NCHOP, &
& BADFRAME, ISEED)
C
GET_SPACE(PARAMS, PHASPACE, LETHARY, BRIGHT, &
E1, E2, NOGravity)
C
For type=90, find source NEUTRON for Monte Carlo, randomly selected in phase
space defined in PARAMS, on the specified surface, with velocity in the
positive Z-direction. (If the surface in the first call is a plane
perpendicular to the Z-axis, it is NOT checked on subsequent calls.)
Function is .TRUE. if neutron source point is within limits of moderator
surface and energy, .FALSE. otherwise. Account for possible gravity-focus
device and/or chopper frequency less than beam repetition rate. For type=
95, read source particles from a direct-access binary file 95; return
FALSE. and NEUTRON.WT = 0 if file wasn't open or error reading file.
A separate entry (GET_SPACE) returns the phase space volume and source
brightness for normalization. If the returned phase space is positive it
is area times solid angle; if negative, just area.  NOTE that GET_SPACE
MUST be called before the first call to N_SOURCE.

P. A. Seeger, May 24, 1994
02 Jul 1994: return neutron weight, even if function is .false. [PAS]
05 Jan 1995: MC_ELMNT.INC now has parameter offsets, not UNIONs [PAS]
10 Jan 1995: use PLNRMTBL for log(E/1meV) instead of PLENEFF; pass
parameters to revised PLTIME; convolute pulse width [PAS]
16 Feb 1995: new calling sequence for PLNRMTBL [PAS]
07 Mar 1995: allow rectangular and off-center phase space apertures [PAS]
22 Mar 1995: omit TOF if parameter negative; omit table lookup if length 1
or less; allow triangular (velocity selector) wavelength [PAS]
28 Mar 1995: source surface in calling sequence, Z not necessarily 0 [PAS]
29 Sep 1995: may call IKDACARP instead of PLTIME for pulse shape [PAS]
10 Dec 1995: set Q and polarization vector to zero [PAS]
25 Jan 1996: if type=95, read direct-access binary source file 95;
add BRIGHT to calling sequence of GET_SPACE [PAS]
06 Jun 1997: changed IKDACARP to PLIKCARP, revised calling sequence [PAS]
04 Sep 1997: second form of phase space if APTR2_R < 0 [PAS,LLD]
04 Nov 1997: square distribution of velocity if fwhm<0; if solid angle=0,
set phase space = -area; index PARAM from 0; use RAN0 [PAS]
09 Nov 1997: correct error in finding source point if not PLANE [PAS]
17 Nov 1997: new calling sequence for RAN0 [PAS]
20 Nov 1997: apply cosine law to statistical weight [PAS]
01 Dec 1997: fix error (since 04 Nov) in square distribution [PAS]
16 Dec 1997: was not computing time for triangle or square pulses [LLD,PAS]
08 Sep 1998: revised call to DIST [PAS]
24 Sep 1998: change all REAL*4 to REAL*8 (except random numbers);
use LAMBDA in time calls [PAS]
20 Oct 1998: restore time after projecting to source surface [PAS]
31 Dec 1998: change min and max energy units from eV to meV [PAS]
24 Jan 1999: eliminate "-1" from value of INDX_E (make parameter E_OFFSET
agree with its definition) [PAS]
06 Feb 1999: set BADFRAME =.FALSE. when no time distribution [PAS]
25 Aug 1999: allow pulse width for monoenergetic source [PAS]
16 Jul 2001: second table format (spline), type 92 [PAS]
19 Jul 2001: change switch function, # spline table entries (type 92) [PAS]
29 Jul 2001: type 92, weight as (avg.lambda)/lambda [PAS]
05 Aug 2001: type 92 switch function with knee [PAS]
17 Dec 2001: define INDX_E before using it [PAS]
14 Mar 2002: allow polarization of simple distribution functions [PAS]
26 May 2002: type 92, decrease BRIGHT if sampled E-range is limited [PAS]
28 May 2002: fix error finding origin of spline tables for time [PAS]
02 Jun 2002: use VZ as component of V0 instead of equal V0 [PAS]
12 Jul 2002: type 92, correct normalization when range limited; added
energy limits to GET_SPACE [PAS]
21 Oct 2002: change MOD() to AMOD() for F90 standard [PAS]
10 Apr 2003: type 95, support multiple readings of source file [PAS]
20 May 2004: move most initialization to GET_SPACE entry [PAS]
23 Dec 2004: change simple distributions from v to lambda, find E1,E2 [PAS]
30 Jul 2006: sum of Maxwellians if type 91, #entries=1, and V0=0 [PAS]
01 Aug 2006: abs(width) in PHASPACE [PAS]
09 Apr 2008: extrapolate if S_EMAX greater than table max (type 92) [PAS]
05 Nov 2009: set constant Polarization for types 91.1, 92.1; turn off
gravity (M=0) if run-time flag was set [PAS]

Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'
INCLUDE 'mc_elmnt.inc'
INCLUDE 'constant.inc'

C Variables in calling sequence:
C    NEUTRON = record with coordinates of neutron (output)
C    PARAMS = array with source parameters (input)
C    SURF = record with surface definition (input)
C    GFACCEL_2 = half the acceleration of 2nd aperture (m/μs/μs, input)
C    NCHOP = chopping ratio, period of chopper / period of beam (input)
C    BADFRAME = flag that neutron belongs to a later proton pulse (output)
C    ISEED = random-number generator seed (input/output)
C    PHASPACE = volume of phase space sampled (m**2-ster) (output)
C    LETHARGY = number of lethargy units of energy space used (output)
C    BRIGHT = luminosity of source (neutrons/ster/m**2/MW/s) (output)
C    E1,E2 = total energy range sampled (meV) (output)
C    NOGravity = flag that gravity has been turned off (input)

RECORD    /PARTICLE/ NEUTRON
RECORD    /SURFACE/  SURF
REAL*8    PARAMS(0:*),GFACCEL_2,PHASPACE,LETHARGY,BRIGHT,E1,E2
INTEGER*4 NCHOP, ISEED
LOGICAL   BADFRAME, GET_SPACE, NOGravity

C Externals:
C     ANGLI     DIST      HUNT8     MOVEX     PLIKCARP  PLMXWLN   PLNRMTBL
C     PLSPLINE  PLTIME    RAN       RAN0      RNDCRCL
REAL*8 DIST
REAL*4 PLIKCARP, PLMXWLN, PLNRMTBL, PLSPLINE, PLTIME, RAN, RAN0

C--- C++
C**********  O  P  E  R  A  T  E  **********
C********** E X I T _ R E G **********
C
SUBROUTINE OPERATE(PART, EXDIST, PARAMS, GEOM, ELEMENT,           &
&                   IREG, JSURF, KSURF, NAME, TRANSMIT, TRACE,     &
&                   FIELDS, FLAG, PART_2, DET_WT, IX, IY, ISEED)

C
C EXIT_REG(PART, GEOM, ELEMENT, IREG, JSURF)
C
C**noPG: sections must be omitted if PGPlot is not available, or null
C routines must be supplied for PGDRAW(PLZ,PLX) and PGMOVE(PLZ,PLX)
C
C Routines to operate on a particle within (or exiting from) a region
C containing material, collimation elements, time-dependent devices,
C samples, or detectors. Included region types/actions are
C
1 amorphous unpolarized material: move to exit w/reduced weight,
  multiple scatter (or absorb) non-transmitted weight
C
2 polycrystalline: move to exit w/reduced weight, scatter rest by
  Bragg or thermal diffuse
C
3 moderators (poly, water, Be), transmit and/or scatter
C  obsolete Be (use type 2.5 instead)
C
6 single-crystal filter: move to exit w/reduced weight
C
7 material w/nuclear resonance(s): move to exit w/reduced weight
C  multi-slit collimator, vertical: selects subregion w/o moving
C  multi-slit collimator, horizontal: selects subregion w/o moving
C  crystal monochromator: reflect from crystal and move to exit;
  rotate coordinates
C
14 toroidal mirror: region divided into subregions by toroidal
  mirror: move to exit w/reduced weight
C  multiple segments with rotation: select subregion, at exit
  transform coordinates and repeat
C  spin flipper; rotate polarization vector and move to exit
blade or disk chopper: move to exit OR select subregion
Fermi chopper: absorption, or move to exit w/reduced weight
removable beamstop: set weight=0 if not transmission mode
fixed-Q or hard-sphere scatterer: transmitted or scattered
particle moves to exit with modified weight
isotropic scatterer with spectrum of energy changes: transmitted
or scattered particle is moved to exit with modified weight
depth inelastic scatter in the impulse approximation: transmitted
or scattered particle is moved to exit with modified weight
reflectometry, multilayer: reflect from surface w/o moving,
same weight if transmitted or reduced weight if scattered
general powder scatterer: transmitted or scattered particle moves
to exit with modified weight
single crystal scatterer: transmitted or scattered particle moves
to exit with modified weight
single detector: determine detection probability
linear detector, vertical: determine y-bin and probability
linear detector, transverse: determine x-bin and probability
2-dimensional detector: determine x- and y-bins and probability
linear detector, longitudinal: determine x-bin and probability
resonant versions of 40-45: determine x- and y-bins and probability
wrapper region for polarization splitting
general types with limited calling sequence
general types with geometry and detectors in calling sequence
source size and phase space: weight=0 if outside surface

17 May 1994: corrected detector encoding for less-than-minimum [PAS]
24 May 1994: added FLAG to calling sequence [PAS]
05 Jan 1995: MC_ELMNT.INC now has parameter offsets, not UNIONS [PAS]
01 Feb 1995: added types 5 (Be), 13 (crystal monochromator), and 35
(reflectometry) [PAS]
17 Feb 1995: added type 31 (hard spheres); more SAVEd variables [PAS]
02 Mar 1995: use negative JSURF as flag for reflection [PAS]
06 Mar 1995: type 31: compute abs(KZ), return trans & refl neutrons [PAS]
09 Mar 1995: types 33 (Ni powder), 44 (longitudinal detector) [PAS]
17 Mar 1995: change POWDER to Ni_POWDR; change argument of PLEXP in
estimating multiple scattering [PAS]
01 Apr 1995: added NAME to calling sequence, type 34 (inelastic) [PAS]
13 Apr 1995: no time binning, eliminate IT from calling sequence; fix
cfactors of 2 in calls to ELSCAT for types 33, 34 [PAS]
08 May 1995: correct errors in multiple scattering [JAG,PAS]
25 May 1995: still was error in inelastic multiple scattering [PAS]
07 Jun 1995: revised handling of monochromator type 13 [MFitz,PAS]
20 Jun 1995: added subtypes 20.n, disk chopper moving vertically and/or
counter-rotating [PAS]
14 Jul 1995: add type 36, powder scatterer [PAS,LLD,UCW]
04 Aug 1995: add type 6, single-crystal filter [PAS]
26 Aug 1995: add type 32, isotropic scatterer; omit type 33 [PAS]
11 Nov 1995: new calling sequence, flag TRANSMIT for transmission mode;
restructured types 30-32,34,36 [PAS]
22 Nov 1995: subtypes 43.n (cylindrical coordinates) [PAS]
15 Feb 1996: scattering probability ~ lambda**2 for types 30,31 [GAO,PAS]
11 Mar 1996: KSURF in calling sequence for output, JSURF input only [PAS]
19 Mar 1996: increment region number by 1 if chopper not open [PAS]
28 Mar 1996: set KSURF = JSURF if particle isn't moved [PAS]
09 May 1996: flag for reflection is KSURF = -KSURF (type 35 only) [PAS]
07 Aug 1996: add type 14, toroidal mirror [PAS,MFitz]
27 Aug 1996: fix error in cylindrical detector phi coordinate [PAS]
07 Jul 1997: if type=43, call DET_2D for new subtype structure [LLD,PAS]
18 Aug 1997: compute JINDEX before starting loop [PAS]
04 Nov 1997: begin PARAMS index at 0; change type 31 to subtype 30.1 [PAS]
07 Jul 1997: if type=43, call DET_2D for new subtype structure [LLD,PAS]
18 Aug 1997: compute JINDEX before starting loop [PAS]
04 Dec 1997: subtype 32.1 [LLD]; KSURF = -JSURF for types 13, 35 [PAS]
13 Jan 1998: restore KSURF = -KSURF for type 35 (error since 04 Dec); fi
type 14 PARAMS calls (error since 04 Nov) [PAS]
23 Apr 1998: dummy calls for region types 70-79 [PAS]
13 May 1998: subtype 32.2, 30.1, 30.2, 34.1, 36.1 [LLD,PAS]
22 Jul 1998: added type 21 (Fermi chopper) [PAS,LLD]
28 Jul 1998: fixed type 32 attenuation, multilevels type 32.2 [PAS]
05 Aug 1998: allow detectors to be encoded in inverse order [PAS]
19 Sep 1998: change REAL*4 to REAL*8 (except random distributions) [PAS]
08 Oct 1998: use DBLE(PART.TOF) in external calls [PAS]
07 Jan 1999: add type 15, multiple segments with rotation; change signs
of bend angles in types 11,12 [PAS]
23 Jan 1999: call to EXIT_REG always returns new IREG [PAS]
06 Mar 1999: change COMPLEX*16 back to COMPLEX*8 for standard F90 [PAS]
10 Mar 1999: fix EXIT_REG for type 15 [PAS]
17 Mar 1999: make factors of PART.WT explicitly REAL*4 [PAS]
02 Apr 1999: inelastic subtype 32.4 with widths [LLD,PAS]
13 May 1999: new calling sequence for monochromator, type 13 [PAS]
14 Jul 1999: type 30: fixed multiple scatter, units of SIGMA_1A [PAS,LLD]
15 Sep 1999: types 11, 12: allow offset and tilt or slope angles, new
subtypes for benders (.1) and radial (.2) [PAS]
18 Sep 1999: add type 7, nuclear resonances (incl. Cd); change calling
sequence of GET_RHO [PAS]
03 Dec 1999: include FIELDS and ELEMENT in calling sequence; call
MOVEB for precession; change calls for 75-59 [PAS]
29 Jan 2000: add 2 diagnostic parameters to MOVEB calls [PAS]
24 Apr 2000: type 13, change sign of axis rotation, 2 new params [PAS]
19 Jun 2000: fix error in crossing slit boundaries for benders [PAS]
22 Jun 2000: type 23, Beamstop MUST be followed by material region [PAS]
28 Jun 2000: new subtype 30.4, 1/Q^2 scatter [PAS]
22 Jul 2000: fix error for crossed Sollers (types 11 AND 12) [PAS]
23 Aug 2000: type 23, fix error (since 22 Jun) incrementing IREG [PAS]
28 Sep 2000: new calling sequence for GET_RHO, including magnetic [PAS]
07 Oct 2000: change WHICHR to function; add ELEMENT to EXIT_REG call [PAS]
08 Oct 2000: rotate polarization whenever coordinates rotate [PAS]
16 Jan 2001: allow unrotated coordinates after a bender (type 11) [PAS]
14 Aug 2001: allow rotation when exiting from type 50 region [PAS]
25 Aug 2001: WHICHR is Logical [PAS]
17 Dec 2001: fix error crossing slits in vertical radial Solier [PAS]
03 Mar 2002: subtype 32.6, multiple scatter with absorption [PAS]
05 Mar 2002: detector may be "Monitor", flag by setting DET_WT<0 [PAS]
13 Mar 2002: add type 16, spin flipper [PAS]
11 Aug 2002: fix error since 4 May crossing slits in types 11,12 [PAS]
19 Aug 2002: new calling sequence for SINGLEV, with polarization [PAS]
20 Oct 2002: type 1, choose between transmit, scatter, absorb [PAS]
14 Nov 2002: new calling sequence for PLQSPHR, including QMAX [PAS]
12 Jan 2003: type 7, include scatter and multiple scatter [PAS]
18 Jan 2003: type 36, standardize treatment of absorption [PAS]
03 Feb 2003: new calling sequence for KERNEL, with scatter fraction [PAS]
14 Jul 2003: new type 37, single crystal sample; type 36, apply secondary
absorption in non-multiple-scattering case [PAS]
25 Aug 2003: type 36, fix error (since 18 Jan) in penetration depth [PAS]
22 Oct 2003: added type 51 for polarization splitting [PAS]
03 Jan 2004: types 2 & 5 (Al and Be) temperature dependence [PAS]
24 May 2004: type 14 (toroidal mirror) had real instead of complex for
C the spin-dependent magnetic term (since 19 Aug 2002) [PAS]
C 09 Mar 2006: types 1-5 split into transmitted & scattered [PAS]
C 05 Sep 2006: type 16, get flip efficiency from velocity ratio [PAS]
C 24 Nov 2007: added type type 33, deep inelastic scatter [PAS]
C 03 Apr 2008: added types 45-49, resonance detectors [PAS]
C 05 Apr 2008: fixed error in type 33 when azimuthal angle limited [PAS]
C 22 Apr 2008: included Doppler broadening for types 45-49 [PAS]
C 20 Mar 2009: C.M. correction for elastic scattering in materials [PAS]
C 22 Mar 2009: added type 3, materials with S(alpha,beta) tables [PAS]
C 18 May 2009: polycrystalline: 2.1=Al, 2.2=diamond/Co, 2.3=Fe, 2.5=Be,
C include multiple scatter as well as transmission [PAS]
C 24 Jun 2009: fixed error when returning only PART_2; added TRACE to
C calling sequence for tracing multiple scatter [PAS]
C 16 Sep 2009: use negative DET_NX,Y,Z as flag for array of tubes [PAS]
C 13 Nov 2009: fix error in type 43 detectors since 16 Sep [PAS]
C
C Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'
INCLUDE 'mc_elmnt.inc'
INCLUDE 'constant.inc'

C Variables in calling sequence:
C PART = record containing description of particle (input/output)
C EXDIST = distance to exit surface particle is aimed at (m) (input/output)
C PARAMS = array with description of what is in this region (input)
C GEOM = structure with all surface and region definitions (input)
C ELEMENT = structure with region space and magnetic parameters (input)
C IREG = region number of device, or subregion within device (input/output)
C JSURF = surface number, if particle is initially on surface (input)
C KSURF = surface number that particle is pointed toward (input/output)
C NAME = name of region, used as file name for type 34 or 37 (input)
C TRANSMIT = flag to compute transmission of sample types 30-39 (input)
C TRACE = 'P' for trace in Plan view, or 'E' for Elevation view (input)
C FIELDS = flag to include magnetic fields (input)
C FLAG = flag set to .FALSE. if (e.g.) chopper in wrong frame (output)
C PART_2 = description of particle created by operation (output)
C DET_WT = statistical weight of detected particle (output)
C IX, IY = position bin numbers of detected particle (output)
C ISEED = random-number generator seed (input/output)
C
RECORD  /PARTICLE/ PART, PART_2
RECORD  /MC_GEOM/ GEOM
RECORD  /MC_ELEMENT/ ELEMENT
REAL*8  PARAMS(0:*), EXDIST, DET_WT
INTEGER IREG, JSURF, KSURF, IX, IY, ISEED
CHARACTER NAME*40, TRACE*1
LOGICAL TRANSMIT, FIELDS, FLAG

C Externals:
C ANGLI ANGTORUS ATTEN_A1 ATTEN_Be ATTEN_Fe ATTEN_X DET_2D
C DIAMOND DISTORUS DTOEX ELSCAT ELSCAT2 FERMI GET_RHO
C GRAV_FOC IMPULSE KERNEL LMONOCR LMORAND LREFLCT MOVEB
C MOVEX NEXTRG ORRAND PdotB PGDRAW PGMOVE PLEXP
C PLNORM PLORENTZ PLQSPHR POWDER RAN REFLAYER RFLN
C SINGLEV SNELL TESTIN WHICHR XCHOPPER
REAL*8  ATTEN_A1, ATTEN_Be, ATTEN_Fe, ATTEN_X, DET_2D, DIAMOND, &
       DISTORUS, GRAV_FOC, REFLAYER, SINGLEV, XCHOPPER
REAL*4  PLEXP, PLNORM, PLORENTZ, PLQSPHR, RAN
COMPLEX*8 GET_RHO
INTEGER NEXTRG
SUBROUTINE OPTICS(PART, GEOM, IREG, ISURF, JREG, &
   RATIO, BETA, ISEED)

Solve for probability of reflection, refraction, or diffuse reflection
for photon PART passing from region IREG across surface ISURF to region
JREG.  Surface parameters are found in GEOM, and the position and
direction in PART are updated.  The ratio of new index of refraction
to old is RATIO.  At exit, JREG will be the new region number for the
particle: IREG if reflected, JREG if transmitted, and 0 if lost or
absorbed.

02 Apr 1985: Changed diffuse reflection to cosine law by calling WOBBLE
[PAS]
05 Apr 1985: Changed diffuse surfaces to include transmission as well as
reflection [PAS]
11 Jun 1987: Added BETA to calling sequence (was always 0.9) [PAS]
03 Feb 1994: converted from COMMON to STRUCTUREs; added PART and GEOM
to calling sequence [PAS]
19 Sep 1998: change all REAL*4 to REAL*8 [PAS]
20 Sep 1998: change calling sequence to WOBBLE (omit BETA) [PAS]

Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'

Variables in calling sequence:
PART = record with particle coordinates (input/output)
GEOM = structure with all surface and region definitions (input)
IREG = current region (input)
ISURF = surface particle is on (input)
JREG = region across ISURF at the particle location; region particle
       will be in after interacting at surface (input/output)
RATIO = (new index of refraction)/(old index of refraction) (input)
BETA = surface roughness parameter (Unused input)
ISEED = random-number generator seed (input/output)
RECORD /PARTICLE/ PART
RECORD /MC_GEOM/ GEOM
REAL*8 RATIO, BETA
INTEGER*4 IREG, ISURF, JREG, ISEED

Externals:
ANGLI, LIGHTRFL, RFLN, SNELL, WOBBLE
LOGICAL LIGHTRFL

REAL*8 FUNCTION PARCYL(X)

c Calculates the value of the the parabolic cylinder function D_-1/2(X)
c for negative values of the argument. The function returns zero if the
c argument is positive.
c
L.L. Daemen, 6 July, 1999
Variables in calling sequence
X = argument of the parabolic cylinder function (input)

Variables in calling sequence:
PART = record containing particle coordinates (input)
GEOM = complete geometry of the simulation (input)
ELEMENT = all element parameters (input)
Pup, Pdown = probabilities of projections of P on B (output)
Direction = unit vector in B direction (or 0 if |B|=0) (output)

Variables in calling sequence:
N = number of square distributions to be convoluted (input)
WIDTHS(N) = array of full widths of square distributions (input)
ISEED = random-number generator seed (input/output)
INTEGER*4 N, ISEED
REAL*8 WIDTHS(N)

C Externals:
C RAN0
REAL*4 RAN0

C--
C++
********** P L E X P **********

REAL*4 FUNCTION PLEXP(ALPHA, PLMAX, ISEED)

Sample a random distribution of the form exp(-ALPHA*X), on the range
(0,PLMAX), or range (0, infinity) if PLMAX=0.

C M.W. Johnson and C. Stephanou, Rutherford Laboratory report RL-78-090.
C 23 Jul 1994: restructured to avoid ALOG(0); test for ALOG(0);
C infinite range if PLMAX is zero [PAS]
C 16 Feb 1995: revise limits for using EXP to range 0.0002 - 17;
C protect against ALPHA=0 [PAS]
C 19 Sep 1998: change all REAL*4 to REAL*8 (except RAN) [PAS]
C 11 Oct 2003: Y > 10^-36 to prevent extremely rare log(0);
C EXP range 0.001-30; save previous factor [PAS]
C
C Variables in calling sequence:
C ALPHA = 1/e factor of exponential distribution (input)
C PLMAX = upper limit on distribution (input)
C ISEED = random-number generator seed (input/output)
IMPLICIT NONE
INTEGER*4 ISEED
REAL*8 ALPHA, PLMAX

C Externals:
C RAN
REAL*4 RAN
C--
C++
********** P L I K C A R P **********

REAL*4 FUNCTION PLIKCARP(LAMBDA, PARAMS, ISEED)

Random time sample from an Ikeda-Carpenter pulse-shape distribution,
of the form of a chi-squared distribution of 6 degrees of freedom,
\[(t/LAMBDA)/TOL0]^2 * exp[-(t/LAMBDA)/TOL0] / 2
\]
Optionally a fraction is convoluted with an epithermal exponential
exp[-(t/LAMBDA)/TAU0]
Also convoluted at long wavelengths with 1 of 2 storage exponentials
exp[-(t/TAU1) or exp[-(t/TAU2)]
with storage probability depending on the value of a switch function
R = exp[-(LAM0/LAMBDA)^POWER] .

C 06 Jun 1997: Changed name (was IKDACARP), changed calling sequence, determine
C TAU locally, allow for additional epithermal exponential [PAS]
C 17 Nov 1997: new calling sequence for RAN0 [PAS]
C 19 Sep 1998: change REAL*4 to REAL*8 (except random distributions);
C ***Calling sequence has LAMBDA in place of E [PAS]
C 07 Mar 2000: TOL0 and TAU0 may be functions of LAMBDA; limits for R [PAS]
Include files with physical constants and parameter list definitions

```
IMPLICIT NONE
INCLUDE 'constant.inc'
#include 'mc_elmnt.inc'
```

Variables in calling sequence

```
C    LAMBDA = neutron wavelength (Å) (input)
C    PARAMS = block containing parameters defining the lineshape (input)
C    ISEED = random-number generator seed (input/output)
    INTEGER*4 ISEED
    REAL*8    LAMBDA, PARAMS(0:*)
```

Externals:

```
C    PLEXP     RAN       RAN0
    REAL*4  PLEXP, RAN, RAN0
```

Sample a Maxwellian energy distribution of mean energy EZERO

```
C    P.A. Seeger, March 25, 1996
C    20 Sep 1998: change REAL*4 to REAL*8 (except random distributions) [PAS]
C    30 Jul 2006: use REAL*8 random numbers, new routine RAN8; new algorithm
```

Variables in calling sequence:

```
C    EZERO = mean energy of distribution (energy units) (input)
C    ISEED = random-number generator seed (input/output)
    IMPLICIT  NONE
    INTEGER*4 ISEED
    REAL*8    EZERO
```

Externals:

```
C    RAN8
    REAL*8    RAN8
```

Sample a normal distribution of mean XBAR and standard deviation SIG

```
C    M.W. Johnson and C. Stephanou, Rutherford Laboratory report RL-78-090.
C    11 Mar 1995: use RAN0 to desequentialize random numbers [PAS]
C    23 Jul 1996: complete rewrite, using Box-Muller transformation, adapted
C    17 Nov 1997: new calling sequence for RAN0 [PAS]
C    20 Sep 1998: change REAL*4 to REAL*8 (except RAN0) [PAS]
C    31 Jul 2006: use RAN8 instead of RAN0, simplify algorithm [PAS]
```

```
IMPLICIT  NONE
#include 'constant.inc'
```
Variables in calling sequence:

- XBAR = mean of normal distribution (input)
- SIG = standard deviation of normal distribution (input)
- ISEED = random-number generator seed (input/output)

```
INTEGER*4 ISEED
REAL*8 XBAR, SIG
```

Externals:

```
REAL*8 RAN8
```

---

**PLNRMTBL**

```
REAL*4 FUNCTION PLNRMTBL(TABLE, NDATA, CENTER, SIGMA, ISEED)
```

Select a random value from a TABLE of length NDATA, whose entry points are equally spaced cumulative Gaussian (or Lorentzian) curve values centered at CENTER in the table with standard deviation (or FWHM if negative) SIGMA. If SIGMA is 0, there is no reference curve and table is sampled uniformly.


13 Feb 1995: CENTER and SIGMA in calling sequence; limits on IR [PAS]
20 Sep 1998: REAL*8 for calling sequence [PAS]
22 Mar 2000: may be Lorentzian or uniform instead of Gaussian reference [PAS]

Variables in calling sequence:

- TABLE = cumulative data points (MUST be monotonic increasing) (input)
- NDATA = number of entries in data table (input)
- CENTER = position in table of center of reference curve (input)
- SIGMA = standard deviation of normal curve (if positive) or FWHM of Lorentzian (if negative) in table index units; zero for no reference (input)
- ISEED = random-number generator seed (input/output)

```
IMPLICIT NONE
INTEGER NDATA
REAL*8 CENTER, SIGMA, TABLE(*)
INTEGER*4 ISEED
```

Externals:

```
REAL*8 PLORENTZ, PLNORM, RAN
```

---

**PLORENTZ**

```
REAL*4 FUNCTION PLORENTZ(XBAR, FWHM, ISEED)
```

Sample a Lorentz (a.k.a. Cauchy) distribution of mean XBAR and full width at half maximum FWHM. The sampled Lorentz distribution is \(a^2/\pi/(a^2 + x^2)\) where \(a\) is the full width at half maximum.


L.L. Daemen, Manuel Lujan Jr. Neutron Scattering Center
October 28, 1997
Variables in calling sequence:

- XBAR = mean of distribution (input)
- FWHM = full width at half maximum of distribution (input)
- ISEED = random-number generator seed (input/output)

```plaintext
IMPLICIT NONE
INTEGER  ISEED
REAL*8   XBAR, FWHM

EXTERNALS:
RAN0
REAL*4 RAN0
```

Function `PLPOISSN`:

```plaintext
REAL*8 FUNCTION PLPOISSN(XM, ISEED)

C Returns a floating-point integer value that is a random deviate drawn from
a Poisson distribution of mean XM. Uses the desquentialized random-
number generator, RAN0.

C 18 Mar 1995: adapted; protect against negative or zero XM [PASeeger]
C 17 Nov 1997: new calling sequence for RAN0 [PAS]
C 19 Sep 1998: change REAL*4 to REAL*8 (except RAN0) [PAS]

C Variable in calling sequence:
XM = mean of Poisson distribution to be sampled (input)
ISEED = random-number generator seed (input/output)
IMPLICIT  NONE
INCLUDE   'constant.inc'
REAL*8    XM
INTEGER*4 ISEED

EXTERNALS:
GAMMLN  RAN0
REAL*8  GAMMLN
real*4  RAN0
```

Function `PLQSPHR`:

```plaintext
REAL*8 FUNCTION PLQSPHR(RADIUS, QMAX, ISEED)

C Select a random momentum transfer Q from the scattering law for a
hard sphere of the specified RADIUS. Units of Q are the inverse of
the units of RADIUS. Probabilities are found from tables of the
cumulative distribution of the Rayleigh scattering law. If QMAX is
not 0, the function repeats till the result is < QMAX.

C 31 May 1985: corrected distribution for solid angle: cumulative distribution
is F = 1 - sin(X)*sin(X)/X**4 + 2*sin(X)*cos(X)/X**3 - 1/X**2
   [A.T.Boothroyd, RAL]
C 21 Jul 1985: Series expansion for F < 0.05 [PAS]
```
Variables in calling sequence

RADIUS = radius of the hard sphere (input)
ISEED = random-number generator seed (input/output)

Implicit None
INTEGER*4 ISEED
REAL*8 RADIUS, QMAX

Variables in calling sequence:
XA = normalized cumulative distribution, monotonic from 0 to 1 (input)
YA = values of the sampled variable corresponding to XA (input)
Y2A = 2nd derivatives of YA at points XA, or may be exponential factor
      in end interval (input)
N = length of arrays (input)
ISEED = random number seed (in/out)

Implicit None
INTEGER N
REAL*8 XA(N), YA(N), Y2A(N)
INTEGER*4 ISEED

Externals
PLEXP RAN
REAL*4 RAN, PLEXP

Select a random time of neutron emission, from a distribution appropriate
  to wavelength LAMBDA (A), described by values in PARAMS. The distribution
  is a sum of an epithermal exponential and a thermal term (linear
  combination of two exponentials) convoluted with a Gaussian. The epithermal
  exponential and the Gaussian constants are proportional to lambda. The
switch function between the epithermal and thermal terms is an exponential
of a power of a wavelength ratio.


10 Jan 1995: all parameters of the distribution provided externally;
new switch function [PAS]
13 Feb 1995: no upper limit to exponential; revised PARAMS [PAS]
27 Mar 1995: protect against missing arguments [PAS]
05 Sep 1995: third exponential term; revised PARAMS [PAS]
19 Sep 1998: change REAL*4 to REAL*8 (except random distributions);
***Calling sequence has LAMBDA in place of E [PAS]
20 May 2004: removed test for pre-1995 code version [PAS]

Include files with physical constants and parameter list definitions
IMPLICIT NONE
INCLUDE 'constant.inc'
INCLUDE 'mc_elmnt.inc'

Variables in calling sequence
LAMBDA = neutron wavelength (A) (input)
PARAMS = block containing parameters defining the lineshape
ISEED = random-number generator seed
INTEGER*4 ISEED
REAL*8    LAMBDA, PARAMS(0:*)

Externals
RAN, PLEXP, PLNORM
REAL*4 RAN, PLEXP, PLNORM

FUNCTION PLTRNGL(XBAR, FWHM, ISEED)
Sample a triangular distribution of mean XBAR and full width
at half maximum of FWHM.

19 Sep 1998: change calling sequence to REAL*8 [PAS]

Variables in calling sequence:
XBAR = center of triangular distribution (input)
FWHM = full width at half maximum (input)
ISEED = random-number generator seed (input/output)
IMPLICIT  NONE
REAL*8    XBAR, FWHM
INTEGER*4 ISEED

Externals:
RAN
REAL*4 RAN

SUBROUTINE POWDER (PARAMS, LAMBDA, SINTH, SIGSCAT, SIGABS, ISEED)
Scattering kernel for isotropic polycrystalline (powder) sample.
Given a powder defined by parameters in PARAMS and the neutron wavelength
LAMBDA, returns the sine of the Bragg angle of the neutron from a randomly
selected reflection, and the macroscopic scattering and absorption cross sections SIGSCAT and SIGABS.

From Uli Wildgruber and Larry Passell, 14 July, 1995; PASeeger.
07 Dec 1995: was always isotropic if lambda > 2nd edge [PAS]
20 Sep 1998: change REAL*4 to REAL*8 (except RAN) [PAS]
31 Dec 1998: change index of PARAMS to 0 [PAS]

Variables in calling sequence:
PARAMS = array with scattering cross section parameters (input)
LAMBDA = neutron wavelength (A) (input)
SINTH = sin(THETA) for chosen reflection (output)
SIGSCAT = macroscopic scattering cross section (/cm) (output)
SIGABS = macroscopic absorption cross section (/cm) (output)
ISEED = Random number seed (input/output)

IMPLICIT NONE
INCLUDE 'mc_elmnt.inc'
INCLUDE 'constant.inc'
REAL*8 PARAMS(0:*), LAMBDA, SINTH, SIGSCAT, SIGABS
INTEGER*4 ISEED

Externals:
HUNT8 RAN
REAL*4 RAN

Variables in calling sequence:
ISEED = 32-bit integer seed value (input/output)
KEYWORD = character for control functions (input)

IMPLICIT NONE
INTEGER*4 ISEED
CHARACTER*1 KEYWORD

External:
RAN
REAL*4 RAN

Random number generator with shuffle table to reduce sequential correlations.
On the first call or if the KEYWORD is 'F' (first), the shuffle table will be reinitialized starting from ISEED. If KEYWORD is 'S' or 'R' then ISEED and the shuffle table will be stored or restored, respectively, and the function will return the same value as the previous call.

Press et al., "Numerical Recipes" (Cambridge, 1986) section 7.1
17 Nov 1997: added Store/Restore capability, KEYWORD in calling sequence [PAS]
19 Nov 1997: protect against fatal error when PREVIOUS = 1 [PAS]

Variables in calling sequence:
ISEED = 32-bit integer seed value (input/output)
KEYWORD = character for control functions (input)

REAL*8 FUNCTION RAN0(ISEED, KEYWORD)

Real*8 random number generator, combining two calls to RAN0.
P. A. Seeger, 30 Jul 2006.
Variables in calling sequence:

ISEED = 32-bit integer seed value (input/output)

IMPLICIT NONE
INTEGER*4 ISEED

External:
RAN0
REAL*4 RAN0

---

SUBROUTINE READ_1D(IUNIT, TYPE, TITLE, FILEID, PFLAG, PARAMS, NP,
&
&       HEADER, NH, X, DX, Y, DY, IFLAG, IERROR)

Search input device IUNIT for the next data file of type TYPE (or next
file of any one-dimensional type if TYPE is blank when called). Return all
header information and data arrays X (which is points if the input value of
PFLAG is 'P' or 'p', or bin boundaries otherwise), DX (which is either bin
normalization if the output value of IFLAG is 0, or the rms of the points
in the bin if IFLAG = 1), Y, and DY (standard deviation). IERROR will be 0
if a suitable data file was found and successfully read, -1 if none found,
and positive if a read error occurred.

The structure of a 1-D file is:
One formatted record (A4,1X,A40,1X,A17) containing TYPE, TITLE, and FILEID.
A block of fewer than 100 parameters, separated by blanks or commas,
and terminated with a '/'. The first parameter must be the number
of channels NCHANS, and if TYPE = 'LOGT' the next four must be the
parameters of the logarithmic time scale. (Additional assignments
of parameters beyond the fifth are assumed by other programs.) The
parameters are not reinitialized; they default to values at input.
A string of up to 720 characters identifying all raw data used and any
operations performed. There are up to nine 80-character records.
The first blank character (after character 1) terminates the string.
One or more "blocks", including one "DATA" block:
The first record of a block is 'POINTS', 'BINS', 'AREA', 'RMSBIN',
'DATA', or 'STDDEV'.
Blocks contain NCHANS (or NCHANS+1 for a BINS block) numbers,
separated by blanks or commas, terminated with a '/'.
If there is no POINTS or BINS block, the bins will default to the
log time scale if TYPE = 'LOGT'; computed linear, logarithmic, or
log/lin/log if appropriate parameters exist in the parameter block;
or integers otherwise.
The STDDEV block or IHIST block (if any) containing respectively
the statistical standard deviations or the number of histories
contributing to each pixel must come after the data block. If
omitted, the defaults will be the square roots of the data.
There may be either an 'AREA' block containing a normalizing function
to be divided into the data, or an 'RMSBIN' block giving the rms
of the X-values used to average within the bin.

Structures other than 1-D files (e.g., 2-D files) may be interspersed
with 1-D files, but must end with a blank record.

08 Dec 1988: changed name (was READ1D); added DX and IFLAG to calling
sequence; read "AREA" or "RMSBIN" blocks.
09 Aug 1989: replace UAHR in calling sequence with PFLAG to return points
instead of bins; compute Q-bins from parameters.

03 Oct 1989: default deviations to 0 if STDDEV block present.

16 Jan 1990: change types 'LOGT' or '<T ' to 't'

18 Jan 1990: revert to 'LOGT' if using parameters to compute bins

19 Jun 1991: change PFLAG to in/out; if blank at input, set to 'P' or 'B'

depending on what is read from file

14 Aug 1992: set PARAMS(18)=3 instead of TYPE='LOGT' for computed bins

26 Jul 1995: compute linear time scale; replace CONTINUEs with END DOs [PAS]

24 Jan 1996: compute log time scale in double precision [PAS]

18 Sep 1997: allow TICK=0 for perfect log scale [PAS]

29 Jul 1998: computed linear scale was wrong (since 1995!) [PAS]

03 Oct 2000: new block type IHIST to compute standard deviation [PAS]

No Externals

Variable in calling sequence:

IUNIT I Input Fortran unit number to be searched and read

TYPE C*4 In/Out Type of data file to search for, or type found

TITLE C*40 Output Title from first record of file

FILEID C*17 Output Instrument_Date_Time identifier

PFLAG C*1 In/Out Conversion flag: 'P' or 'p' to return X as

points, or ' ' to return whatever was recorded

in file and then set to 'B' or 'P'

PARAMS R(*) Output Parameters saved with the data file;

PARAMS(1) = NCHANS = number of channels

PARAMS(2..5) = log time parameters

PARAMS(18..23) = Q-binning parameters

NP I Input Number of PARAMS to be returned to caller

HEADER C(9)*80 Output Character string describing operations

previously performed on data

NH I Output Number of characters in HEADER

X R(*) Output Bin boundaries (NCHANS+1 numbers), or points

(NCHANS numbers) if PFLAG is 'P' or 'p'.

DX R(*) Output Normalizing factor (area) of each bin, or

rms of points used in average (see IFLAG)

Y R(*) Output Data values summed or averaged over bins

DY R(*) Output Standard deviations of Y

IFLAG I Output 0 if DX is normalizing factor, 1 if rms, or

-1 if neither was read from file

IERROR I Output 0 for successful read, -1 for end-of-file,

or system-dependent message number

--

C
C********** R E A D _ 2 D **********

C

SUBROUTINE READ_2D(IN, TYPE, TITLE, FILEID, PFLAG, PARAMS, NP, &

& HEADER, NH, X, DX, Y, DY, NNY, &

& Z, DZ, NXZ, IFLAG, IERROR)

C

Search input device IN for the next data file of type TYPE (or next file
C of any two-dimensional type if TYPE is blank when called). Return all header
C information, the one dimensional arrays X, DX, Y, and DY, and the 2D data
C arrays Z(X,Y) and DZ(X,Y), stored in arrays with first dimension NXZ. The
C X and Y arrays are normally bin boundaries, but may be returned as points by
C setting PFLAG; if "PX" then X will be points, if "PY" then Y will be points,
C and if "P" both. The numbers of X and Y bins are always read from the header
C (1st and 24th entries in PARAMS). The input NY may be reduced to the value
C in the data file. DX and DY may be either normalizing functions (areas of
C bins) or rms deviations of points included in bin averages (flagged by the
C value returned in IFLAG: 1 if DX is rms, 2 if DY is rms, 3 if both, -1 if
no such blocks were read). DZ contains the standard deviations of Z, either as read from the input file or assuming Poisson statistics. At exit, IERROR will be 0 if a suitable data file was found and successfully read, -1 if no file found, and positive if a read error occurred.

The structure of a 2D file is:
One formatted record (A4,1X,A40,1X,A17) containing TYPE, TITLE, and FILEID A block of fewer than 100 parameters, separated by blanks or commas, and terminated with a '/'. The first parameter must be the number of channels NX in the X direction. (Additional assignments of parameters are listed below.) The parameters are not reinitialized; they default to values at input.
A string of up to 720 characters identifying all raw data used and any operations performed. There are up to nine 80-character records. The first blank character (after character 1) terminates the string.
One or two optional blocks, either "POINTS" or "BINS", the first defining the X-axis and the second (if any) defining the Y-axis. If the blocks are not present, X (or Y) will default to the log t scale if the first (or any subsequent) letter of TYPE is 'T', or to integer steps otherwise.
One or two optional "AREA" or "RMSBIN" blocks, containing normalization factors (if the keyword is "AREA") to be divided into the raw histogram counts, or else the rms average ("RMSBINS") of the values of X or Y used in computing Z averages. The first such block refers to X, and the second (if any) to Y. By using "AREA" blocks, Z may be stored in the unnormalized form to preserve its Poisson statistics. These blocks are returned to the caller as DX and DY, which will be set to unity if the blocks are not present.
Any number (at least one!) of rows of Z:
The first record of a row block has a '=' within the first 4 characters and a value of IY anywhere in the 15 characters following. The value of IY must be between 1 and NY, or within the time-slice limits given in the header if Y is time.
A block of NX values, separated by blanks or commas, terminated with a '/'.
Each row block may be immediately followed by a "STDDEV" block containing NX values of the standard deviation of Z or by an "IHIST" block with number of histories for each pixel; otherwise Z will be assumed Poisson and the standard deviation will be the square root.

Structures other than 2D files (e.g., 1D files) may be interspersed with 2D files, but must end with a blank record.

Definitions of the parameter block are
PARAM(1) = NX, number of points in a block, or bins in X-direction
PARAM(2) = DTOVERT, dt/t for log time scale
PARAM(3) = TICK, resolution of master clock in Fastbus
PARAM(4) = TSTART, beginning of first time slice
PARAM(5) = TZERO, time delay of detector
PARAM(6) = maximum time, last bin boundary
PARAM(7) = size of detector element in X-direction (mm)
PARAM(8) = size of detector element in Y-direction (mm)
PARAM(9) = X of beam centerline (detector units)
PARAM(10) = Y of beam centerline (detector units)
PARAM(11) = power of the units of Z
PARAM(12) = source-to-sample distance (m)
PARAM(13) = sample-to-detector distance (m)
PARAM(14) = rms X of the beam spot (mm, see also PARAM(31))
PARAM(15) = minimum detector radius included
PARAM(16) = IFIRST, first time slice included
PARAM(17) = last time slice included
PARAM(18) = Q-bin type: 0 = linear, 1 = logarithmic, 2 = log/lin/log
or 3 = time bins with Fastbus algorithm
PARAM(19) = switch point from log to linear
PARAM(20) = switch point from linear back to log
PARAM(21) = number of bins between Q1 and Q2
PARAM(22) = minimum Q bin boundary
PARAM(23) = maximum Q bin boundary
PARAM(24) = NY, number of blocks of data, or bins in Y-direction
PARAM(25) = integrated source power in MW-s
PARAM(26) = resolution width factor, dR/R < (this factor)*dQ/Q
PARAM(27) = detector efficiency at 1 A
PARAM(28) = scattering angle to a fixed detector (degrees)
PARAM(29) = incident energy, direct-geometry inelastic (meV)
PARAM(30) = final energy, inverted-geometry inelastic (meV)
PARAM(31) = rms Y of the beam spot (mm, see also PARAM(14))
PARAM(32) = T_ZERO, number of pulses in a TOF experiment
PARAM(33) = TAU, parameter in dead-time correction (us)

02 Nov 1988: add DZ to calling sequence; read STDDEV blocks.
08 Nov 1988: determine NY; set bins to 0 before reading.
08 Dec 1988: renamed (was READ2D); revised calling sequence!!!
allows RMSBIN blocks.
21 Dec 1988: IFLAG=-1 if no AREA or RMSBINS blocks
12 Aug 1989: use new header index information; replace UAHR in calling
sequence with PFLAG for points instead of bins
25 Sep 1989: offset IY if Y is time and IFIRST is given in header
05 Jul 1992: also offset IY if Y is wavelength
24 Jun 1994: MS Fortran version recompiled with "[HUGE]" attributes
13 Mar 1995: output integer BINS if input block was null [PAS]
29 Mar 1995: restructure to speed up when dimensions large; require
PARAM(24); allow 't' as well as 'T' in TYPE [PAS]
06 Apr 1995: eliminate BACKSPACE when reading '=' blocks [JAG,PAS]
26 Jul 1995: compute linear time bins; use END DO, etc. [PAS]
21 Sep 1995: correct error (since 6 Apr), not reading files without
explicit STDDEV blocks [PAS]
05 Jan 1996: fix NH too small by 1 (since 26 July) [PAS]
24 Jan 1996: compute log time scale in double precision [PAS]
18 Sep 1997: allow TICK=0 for perfect log scale [PAS]
03 Oct 2000: new block type IHIST to compute standard deviation [PAS]
29 May 2006: fixed error: must restore input NY when skipping data [PAS]
29 Dec 2008: fixed error: TMAX for linear Y time bins was stored in X
instead of Y, fatal if NY>NX [PAS]

No Externals

Variable in calling sequence:
IN I Input Fortran unit number to be searched and read
TYPE C*4 In/Out Type of data file to search for, or type found
TITLE C*40 Output Title from first record of file
FILEID C*17 Output Instrument_Date_Time identifier
PFLAG C*2 Input Flag to return points instead of bins;
'PX' for X, 'PY' for Y, or 'P' for both
PARAMS R(*) In/Out Parameters saved with the data file;
PARAMS(1) = NX = number of channels
PARAMS(2..5) = log time parameters
NP I Input Number of PARAMS to be returned to caller
HEADER C(9)*80 Output Character string describing operations
previously performed on data
C   NH   I       Output   Number of characters in HEADER
C   X    R(*)   Output   Bin boundaries (NX+1 numbers)
C   DX   R(*)   Output   Normalization factor to be divided into Z
C                              as a function of its first index, OR rms of
C                              bin (see IFLAG)
C   Y    R(*)   Output   Bin boundaries for the second index of Z
C   DY   R(*)   Output   Normalization factor to be divided into Z
C                              as a function of its second index, OR rms of
C                              bin (see IFLAG)
C   NNY  I       In/Out   Maximum number of Y blocks allowed/returned
C   Z    R(NXZ,*), Output  2D data array
C   DZ   R(NXZ,*) Output   Standard deviations of Z
C   NXZ  I       Input    First dimension of Z and DZ in calling program
C   IFLAG I       Output   1 if DX is rms widths of X-bins, 2 if DY is
C                              rms width of Y-bins, 3 if both, 0 if neither;
C                              -1 if no blocks were read
C   IERROR I       Output   0 for successful read, -1 for end-of-file,
C                              or system-dependent message number
C--
C++
C********* R E A L 4 O U T **********
C
SUBROUTINE REAL4OUT(Q, N, IUNIT)
C  Output an array of N real*4 numbers Q to unit IUNIT, as ASCII characters
C  in free format with between 5-1/2 and 7 significant digits, separated
C  by commas, with 126 or fewer characters per record. Each unit record
C  begins with a blank, and the entire array is terminated with a '/'.
C
P. A. Seeger, Los Alamos National Laboratory, April 9, 1987.
C  30 Sep 1987: Use "\," instead of "0,\," for zero.
C  24 Nov 1987: Changed lower limit for F format.
C  05 Dec 1987: Decreased digits in F format by one.
C  02 Dec 1988: Eliminate trailing ",\,"; put null after "/".
C  23 Sep 1989: omit final null if buffered line too long.
C  12 Feb 1991: omit "E" and lead "0" in exponent of E format; raise
C                  lower limit for "F" format
C  30 Apr 1992: omit trailing 0s in E format
C  13 Mar 1995: omit null after "/" [PAS]
C  16 Aug 1995: put "E" back (so IDL can read data) [PAS]
C  02 Jan 1996: make string long enough to hold "E-nn," [RvD,PAS]
C  14 Sep 1998: change name to show single precision; "0,\," [PAS]
C
C  Externals:
C  DIGITS4
C
C  Variables in calling sequence
C  Q    R(*)   Input   Array of real*4 numbers to be output
C  N    I       Input   Number of numbers
C  IUNIT I       Input   Logical unit number for output
C--
C++
C********* R E A L 8 O U T **********
C
SUBROUTINE REAL8OUT(Q, N, IUNIT)
C  Output an array of N real*8 numbers Q to unit IUNIT, as ASCII characters
C  in free format with between 9-1/2 and 11 significant digits, separated
C  by commas, with 126 or fewer characters per record. Each unit record
C  begins with a blank, and the entire array is terminated with a '/'.
EXTERNALS:

**DIGITS**

**Variables in calling sequence**

- **Q** real*8, Input. Array of real*8 numbers to be output.
- **R(*)** real*8, Input. Array of real*8 numbers to be output.
- **N** integer, Input. Number of numbers.
- **IUNIT** integer, Input. Logical unit number for output.

---

**R F L A Y E R**

**REAL*8 FUNCTION REFLAYER(KZERO, PARAMS)**

C

Calculate the probability of reflection of a neutron from any number of
layers on top of a substrate. The wavevector component perpendicular
to the surface is **KZERO**, and the layers are defined in **PARAMS**. In case
of error, the probability is set to -1. If the external medium is not
void, its value of 4pi*n*b should be subtracted from the value for each
layer.

Greg Smith 1/30/95

---

**R F L N**

**SUBROUTINE RFLN(PART, COSTH, AP, BP, CP)**

C

Perform reflection of particle **PART**, with angle of reflection
equal to angle of incidence (cosine of angle = **COSTH**, measured with
respected to surface normal $AP, BP, CP$). Sign of $COSTH$ is changed to represent new direction of particle.

M. W. Johnson, Rutherford and Appleton Laboratories report RL-80-065.

Modified calling sequence, P. A. Seeger, 1984.

03 Feb 1994: converted from COMMON to STRUCTUREs; add PART to calling sequence [PAS]

20 Sep 1998: change all REAL*4 to REAL*8 [PAS]

Definitions of STRUCTUREs:

- IMPLICIT NONE
- INCLUDE 'mc_geom.inc'

Variables in calling sequence:

- PART = record with particle coordinates
- $COSTH$ = cosine of angle of incidence (input)
- $AP, BP, CP$ = unit vector in direction of normal to surface (input)

```plaintext
RECORD /PARTICLE/ PART
REAL*8 $COSTH, AP, BP, CP$
```

No externals

------

Subroutine RK4BLOCH($T$, $P$, $B0$, $DT$, $Pout$, $Bfunction$, $eps$, $PART$, $PARAMS$, $NAME$, $S$)

Fourth-order Runge-Kutta stepper specific to solving Bloch's precession equation. Given the 3-vector $P$ (representing polarization) at an initial time $T$ (us), the initial value of the magnetic induction $B0$ (T), and a routine $Bfunction$($T$, $B$, $PART$, $PARAMS$) that returns $|B|$ and $B$ as a function of time; use a modified fourth-order Runge-Kutta method to advance the solution of Bloch's equation over a time step from $T$ to $T+DT$ (us) and return the rotated vector as $Pout$ (may be same array as $P$). The total line integral of $-\gamma |B| dt$, computed by Romberg integration, is returned as $S$.

Runge-Kutta from Press, Flannery, Teukolsky, & Vetterling, "Numerical Recipes" (Cambridge University Press, 1986), sec. 15.1

RK4 adapted by P. A. Seeger, Los Alamos National Laboratory, Aug. 30, 1989

29 Jan 2000: complete revision, convert to angular variables instead of linear extrapolation; Bloch equation explicit [PAS]

31 Jan 2000: Romberg integration for $S$ [PAS]

11 Feb 2000: added $NAME$ to call, for look-up table files [PAS]

12 Feb 2000: improve treatment of $P$ parallel to $B$ [PAS]

17 Mar 2002: limit when $B$ is close to $-Z$ direction; punt if $|B0|=0$ [PAS]

Variables in calling sequence:

- $T$ = neutron time-of-flight at starting point (us) (input)
- $P$ = initial value of the polarization vector (input)
- $B0$ = magnetic induction vector at beginning of step (T) (input)
- $DT$ = length of integration step (us) (input)
- $Pout$ = computed value of $P$ at ($T+DT$) (output)
- $Bfunction$ = user-supplied external function name (input)
- $eps$ = required absolute accuracy for each component of $P$
- $PART$ = particle structure (position & velocity) for $Bfunction$ (input)
- $PARAMS$ = array of parameters for $Bfunction$ (input)
- $NAME$ = file name for look-up tables for $Bfunction$ (input)
- $S$ = line integral of $-\gamma |B| dt$ (rad) (output)
IMPLICIT NONE
INCLUDE 'constant.inc'
REAL*8 T,P(3),B0(3),DT,Pout(3),Bfunction,eps,PART(*),PARAMS(*),S
CHARACTER NAME*40

EXTERNALS:
Bfunction BINTEGRAL
EXTERNAL Bfunction
REAL*8 BINTEGRAL

LOGICAL FUNCTION RKPRECES(T, P, Htry, eps, Hdid, Hnext, &
& Bfunction, PART, PARAMS, NAME, S)

Step manager for fourth-order Runge-Kutta specific to solving Bloch's
precession equation. Monitors local truncation error to ensure accuracy, and
adjusts stepsize. The user-supplied function Bfunction(T, B, PART, PARAMS,
NAME) is used to compute the magnetic induction vector B and |B|. RKPRECES
returns .FALSE. if the stepsize is reduced to less than 12-bit significance.
The integrated precession is returned as S.

Runge-Kutta from Press, Flannery, Teukolsky, & Vetterling, "Numerical
Recipes" (Cambridge University Press, 1986), sec. 15.2
12 Jan 2000: complete revision, change to logical function; modified for
coefficients of precession instead of linear extrapolation;
calls explicit step routine for Bloch's equation. [PAS]
02 Feb 2000: if stepsize had to be reduced, don't increase as much [PAS]
11 Feb 2000: added NAME to call, for table look-up files [PAS]
14 Mar 2002: omit B from calling sequence; give up when H gets small [PAS]
17 Mar 2002: protect against zero-field point [PAS]
08 Jul 2006: put NAME in 2 more calls to Bfunction [PAS]

Variables in calling sequence:
T = neutron time-of-flight at beginning of step; replaced by T+Hdid
at exit (us) (input/output)
P = initial/final 3-vector polarization (input/output)
Htry = trial value for stepsize (us) (input)
eps = required absolute accuracy for each component of P; note that
    execution time increases rapidly as eps gets smaller (input)
Hdid = stepsize actually achieved (us) (output)
Hnext = estimated next stepsize (us) (output)
Bfunction = user-supplied external function (input)
PART = particle structure (position & velocity) for Bfunction (input)
PARAMS = array of parameters for Bfunction (input)
NAME = file name for look-up tables for Bfunction (input)
S = line integral of -gamma |B| dt (rad) (output)

IMPLICIT NONE
REAL*8 T,P(3),Htry,eps,Hdid,Hnext,Bfunction,PART(*),PARAMS(*),S
CHARACTER NAME*40
EXTERNAL Bfunction

EXTERNALS:
RK4BLOCH
SUBROUTINE RNDCRCL(X, Y, ISEED)

Generate random point (X,Y) within a unit circle

11 Mar 1995: use RAN0 to desquentialize random numbers [PAS]
17 Nov 1997: new calling sequence for RAN0 [PAS]
20 Sep 1998: change REAL*4 to REAL*8 (except RAN0) [PAS]

Variables in calling sequence:
X, Y = random point (output)
ISEED = random-number generator seed (input/output)

IMPLICIT NONE
REAL*8 X, Y
INTEGER*4 ISEED

Externals
RAN0
REAL*4 RAN0

LOGICAL FUNCTION RSPLINT(XA, YA, Y2A, N, Y, I, R, X, dYdX)

Subroutine to reverse interpolate a value Y in a spline table (XA,YA)
with 2nd derivatives Y2A, of length N. If a result is found the
function will be .TRUE., I will be the index number of the cell
containing Y, R will be the ratio of Y in the width of the cell,
X is the interpolated result, and dY/dX is the 1st derivative at
that point (only one root of the cubic will be returned). The input
array YA must be monotonic increasing, with no repeated values.
Note: the table is assumed to be a cumulative distribution, and the
underlying pdf is assumed to be peaked and to decrease monotonically
at both ends; a 2nd derivative with the "wrong" sign is used as a
flag to treat the end intervals as exponentials.

13 Dec 2004: exponential in end cells if Y2A has "wrong" sign [PAS]

Variables in calling sequence:
XA = independent variable of the spline curve (input)
YA = array in which to interpolate (input)
Y2A = 2nd derivatives of YA with respect to X(input)
N = length of arrays YA and Y2A (input)
Y = value to be found in table (input)
I = index of table value next lower (or =) YA (output)
R = ratio of location of Y within cell (output)
X = interpolated abscissa (output)
dYdX = 1st derivative at interpolated point (output)

IMPLICIT NONE
INTEGER*4 N, I
REAL*8 XA(N), YA(N), Y2A(N), Y, R, X, dYdX

No externals

LOGICAL FUNCTION RSPLINT(XA, YA, Y2A, N, Y, I, R, X, dYdX)
LOGICAL FUNCTION SAMESURF(SURF1, SURF2, RATIO)

Tests whether two surface definitions are the "same"; that is, whether
they have the same significant coefficients and the sum of the deviations
from unity of the ratios of the coefficients is less than 1.E-6. The
ratio of the coefficients (SURF1 / SURF2) is returned as RATIO; note
that if RATIO < 0, the sense of the surfaces is opposite.

P.A. Seeger, 12 Sep 1998
22 Jan 1999: change criterion from 10^-6 to 10^-8 [PAS]
10 Apr 1999: change back to 10^-6 [PAS]
31 Aug 2002: add test of BETA, change from .NEQV. to .XOR. [PAS]

Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'

Variables in calling sequence:
SURF1, SURF2 = the two surface structures to be compared (input)
RATIO = ratio of coefficients, SURF1 / SURF2 (output)
RECORD /SURFACE/ SURF1, SURF2
REAL*8 RATIO

No externals

********** S I N G L E V **********

REAL*8 FUNCTION SINGLEV(LAMBDA, ITYPE, JPOLAR, PARAMS, SIGTOT, &
& SIGSCAT, REALCOH, IMAGCOH)

Calculates slow neutron cross-sections and complex scattering length from a
generalized 1-channel R-matrix (isolated s-wave resonances). Resonant and
non-resonant contributions from any number of spin-isotope combinations are
included. Examples are Cd, Sm, or 240Pu.

Eric Lynn, May 26, 1988, revised 21 Jul 1988 [PAS]; see J.E.Lynn & P.A.Seeger,
Atomic Data and Nuclear Data Tables 44 (1990) 191-207.

18 Sep 1999: pass parameters in call instead of common; computed quantities
in call (not just ITYPE); change sign of IMAGCOH; omit full
R-matrix option; macroscopic instead of microscopic x-sections;
real*8; save previous case [PAS]
18 Aug 2002: revise for polarized samples, added JPOLAR to call [PAS]
24 Jun 2009: make test of previous case single-precision [PAS]

Variables in calling sequence
LAMBDA, neutron wavelength (Å) (input)
ITYPE, flag for quantity returned as function value: 0=SIGTOT,
1=REALCOH, 2=SIGSCAT, 3=macroscopic absorption cross section (/m),
4=modulus of coherent scattering-length density (10^10/cm^2),
5=IMAGCOH, 6=reduced total cross section (eV^1/2 /m)
JPOLAR, flag for spin: -1 for s=j-1/2, 0 for j=0, +1 for s=j+1/2 (input)
PARAMS, block of resonance parameters (input)
SIGTOT, total macroscopic cross section (/m) (output)
SIGSCAT, macroscopic scattering cross section (/m) (output)
REALCOH, Re(coherent scattering-length density) (10^10/cm^2) (output)
IMAGCOH, Im(coherent scattering-length density) (10^10/cm^2) (output)
IMPLICIT NONE
REAL*8 LAMBDA, PARAMS(0:*), SIGTOT, SIGSCAT, REALCOH, IMAGCOH
INTEGER ITYPE, JPOLAR
INCLUDE 'constant.inc'
INCLUDE 'mc_elmnt.inc'
C
C No Externals
C--
C++
C********** S N E L L **********
C
SUBROUTINE SNELL(PART, COSTH, AP, BP, CP, RATIO)
C
C Perform refraction of particle PART, with \( \sin(\text{angle of refraction}) = \frac{\sin(\text{angle of incidence})}{\text{RATIO}} \). If RATIO < 1 and the angle exceeds
C the critical angle, the particle is reflected. The surface normal C is (AP,BP,CP) and the cosine of the angle of incidence is COSTH;
C at exit, COSTH is changed to the new angle.
C
C 03 Feb 1994: converted from COMMON to STRUCTUREs; add PART to calling C sequence [PAS]
C 08 Sep 1997: internal computations in double precision [PAS,LLD]
C 20 Sep 1998: change calling sequence to REAL*8 [PAS]
C
C Definitions of STRUCTUREs:
    IMPLICIT NONE
    INCLUDE 'mc_geom.inc'
C
C Variables in calling sequence:
C    PART = record with particle coordinates (input/output)
C    COSTH = cosine of angle of incidence (input/output)
C    AP,BP,CP = unit vector in direction of normal to surface (input)
C    RATIO = ratio of indices of refraction, new/old (input)
    RECORD /PARTICLE/ PART
    REAL*8 COSTH, AP, BP, CP, RATIO
C
C External:
C    RFLN
C--
!++
!**********  S N G L X T A L  **********
!
SUBROUTINE SNGLXTAL(PART, PARAMS, SC_file, abs_xlen, tot_xlen, &
 & ISEED)
!
ENTRY SC_SCATTER(PART, COHERENT, ISEED)
!
! Single crystal with mosaic. Delta-D/D option for finite-size effects.
! The mosaic may EITHER be specified isotropic by setting the mosaic input
! parameter, OR anisotropic by providing a mosaic matrix.
! Crystal structure is specified with an ascii data file. The first three
! numbers on each line are the \((h,k,l)\) indices of the reciprocal lattice point,
! and the 7th number is the value of the structure factor \(|F|^2\), in barns;
! may be output of the Crystallographica program.
!
! Written by: Kristian Nielsen, Risø National Laboratory, December 1999,
! for McStas neutron ray-tracing package.
! Complete description in report Risø-R-1175(EN), pp. 72-79.
! http://neutron.risoe.dk/documentation/manual/mcstas-1.4-manual.ps.gz
!
!Copyright (c) Risø National Laboratory.
!All rights reserved.
!Redistribution and use in source and binary forms, with or without
!modification, are permitted provided that the following conditions
!are met:
!1. Redistributions of source code must retain the above copyright
!   notice, this list of conditions and the following disclaimer.
!2. Redistributions in binary form must reproduce the above copyright
!   notice, this list of conditions and the following disclaimer in the
!   documentation and/or other materials provided with the distribution.
!3. All advertising materials mentioning features or use of this software
!   must display the following acknowledgement:
!   This product includes software developed by Risø National
!   Laboratory and its contributors.
!4. Neither the name of the Laboratory nor the names of its contributors
!   may be used to endorse or promote products derived from this software
!   without specific prior written permission.
!THIS SOFTWARE IS PROVIDED BY RISØ AND CONTRIBUTORS ``AS IS'' AND
!ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
!IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
!ARE DISCLAIMED. IN NO EVENT SHALL RISØ OR CONTRIBUTORS BE LIABLE
!FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
!DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
!OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
!HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
!LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
!OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
!SUCH DAMAGE.
!
! 14 Jul 2003: translated from C++ to Fortran90, new code generally in
! Upper Case; McStas ---> NISP, made into 2 procedures (PAS)
! 25 Jul 2003: full matrix for anisotropic mosaic (PAS)
! 05 Aug 2003: fix my error in V0, McStas error in e2_x (PAS)
! 20 May 2004: enter with ISEED=0 to reinitialize (PAS)
!
! Definitions of NISP STRUCTUREs:
! IMPLICIT NONE
! INCLUDE 'mc_geom.inc'
! INCLUDE 'mc_elmnt.inc'
! INCLUDE 'constant.inc'
!
! Variables in calling sequences:
! PART = record containing description of particle (input/output)
! PARAMS = array with description of what is in this region (input)
!          (see file mc_elmnt.inc for details)
! SC_file = name of region, used as data file name for this type (input)
! abs_xlen = macroscopic absorption cross section (/cm) (output)
! tot_xlen = macroscopic total cross section (/cm) (output)
! COHERENT = flag that coherent scattering occurred (output)
! ISEED = random-number generator seed (input/output)
! RECORD /PARTICLE/ PART
! REAL*8 PARAMS(0:*), abs_xlen, tot_xlen
! CHARACTER SC_file*40
! LOGICAL COHERENT
! INTEGER ISEED
!
! Externals
! norm_vec ORRAND PLNORM RAN vecProd
! REAL*4 PLNORM,RAN
!--
SUBROUTINE SOLENOID(X, Y, Z, PARAMS, B)

Magnetic induction at point (X,Y,Z) produced by a solenoid with parameters
given in PARAMS. The induction is returned in the 3-vector B. The algorithm
for vector potential of a solenoid is from E.Durand, "Magnetostatique,"
(Masson, Paris, 1968) pp.97-100. In the coordinate system of the solenoid,
A(rho,z) = factor*[G(rho,z;-b) - G(rho,z;b)],
where G(rho,z,zeta) = 2a(z-zeta)/R *[E(k)-K(k)]/k^2 - (1+n)/n [PI(n,k)-K(k)]
and R=sqrt((a+rho)^2+(z-zeta)^2), k=2 sqrt(a rho)/R, n=-4a rho/(a+rho)^2
c and K(k),E(k),PI(n,k) are complete elliptic integrals of 1st, 2nd, & 3rd
kinds.
The direction of A is circular around the solenoid axis, so B=curlA gives
Brho = - factor*[dG/dz(b) - dG/dz(-b)]
Bz = factor/rho [d(rho G)/drho(b) - d(rho G)/drho(-b)]
That is, the partial derivatives of G must be evaluated at the two ends of
the solenoid. Durand gives dG/dz but NOT d(rho G)/drho. We have found an analytic
expression using complete and incomplete elliptic integrals as found in
J.Spanier & K.B.Oldham, "An Atlas of Functions" (Hemisphere, 1987), secs. 61-
62.
The result is that Brho uses only 1st & 2nd kinds, while Bz uses 1st & 3rd
kinds.
To avoid singularities, the thickness of the conducting surface is assumed to be
0.0005 times the radius.

*** didn't give correct Bz very far off axis!!!
27 Mar 2000: rectangular coordinates, axis orientation parameters [PAS]
28 Mar 2000: revised call to CMPLT_PI to return K and E [PAS]
13 Dec 2001: revised call for new algorithm in CMPLT_PI [PAS]
15 Dec 2001: special case for very far (magnetic dipole) [PAS]
17 Jan 2002: new expression for Bz, correct for all rho [PAS]
07 Apr 2003: clean up a little [PAS]

Variables in calling sequence
X,Y,Z = coordinates where the induction is to be calculated (m) (input)
PARAMS = parameters of the region containing the solenoid (input)
B = magnetic induction vector in lab coordinate system (T) (output)

IMPLICIT NONE
INCLUDE 'constant.inc'
INCLUDE 'mc_elmnt.inc'
REAL*8 X, Y, Z, PARAMS(0:*), B(3)

C Externals:
C CMPLT_PI
REAL*8 CMPLT_PI

C** ******** SRC_PROB ********
REAL*8 FUNCTION SRC_PROB(E, T, PARAMS, TABLE, PROB_E)

Find probability density in energy (per meV) and time (per us) for
source particle emission from energy TABLE and time function in PARAMS.
Result dimensions are neutrons/(MW-s)/ster/mm^2/meV/us. If T<0,
compute PROBE but return 0 for SRC_PROB.
Variables in calling sequence:

- \( E \) = neutron energy (meV, input)
- \( T \) = emission time (us, input)
- PARAMS = type 91 (or 92) parameter block defining table structure and analytic form for emission time (input)
- TABLE = table of cumulative energy distribution (weighted by \( 1/E \)) of source spectrum on equally spaced normal-curve values of \( \log_{10}(\text{energy}/1 \text{ meV}) \) –OR– spline function table of cumulative energy distribution (weighted by \( 1/\lambda \)) of source spectrum at 33 implicit values of the normalized sum
- NOTE: either form of energy table may be followed by spline tables for the time of emission.
- PROB_E = neutrons/(MW-s)/uster/mm^2/meV, integrated over time (output)

IMPLICIT NONE
INCLUDE 'mc_elmnt.inc'
INCLUDE 'constant.inc'
REAL*8   E, T, PARAMS(0:*), TABLE(*), PROB_E

Externals:
INTERP8    RSPLINT
LOGICAL  RSPLINT

**TESTIN**(PART, GEOM, IREG, ISURF)

Tests whether the particle defined in PART is inside region IREG as defined in GEOM. If ISURF is not 0, then that surface is not included in testing, in order to avoid roundoff/truncation errors when the particle is known to be on the surface. Function is .TRUE. if inside region IREG, .FALSE. if not. No parameters are changed.

M.W. Johnson and C. Stephanou, Rutherford Laboratory report RL-78-090.
Modified by P.A. Seeger (1980) for case of particle on a bounding

03 Feb 1994: converted from COMMON to STRUCTUREs; added PART and GEOM to calling sequence [PAS]

29 Jun 1995: don't test surface if STYPE >10 [PAS]

04 Apr 1996: double precision; local SURF and (X,Y,Z) [LLD,PAS]

Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'

Variables in calling sequence:
PART = record with particle coordinates (input)
GEOM = structure with all surface and region definitions (input)
IREG = number of region being tested for interiorness (input)
ISURF = number of surface that particle PART is on (input)

RECORD /PARTICLE/ PART
RECORD /MC_GEOM/ GEOM
INTEGER*4 IREG, ISURF

No externals

C********** TYPE _ 7 0 **********

SUBROUTINE TYPE_70(PART, EXDIST, PARAMS, NAME, TRANSMIT, FLAG, &
PART_2, ISEED)

Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'

Variables in calling sequence:
PART = record containing description of particle (input/output)
EXDIST = distance to exit surface particle is aimed at (m) (input/output)
PARAMS = array with description of what is in this region (input)
NAME = name of region, used as file name for type 34 (input)
TRANSMIT = flag to compute transmission of sample types 30-39 (input)
FLAG = flag set to .FALSE. if (e.g.) chopper in wrong frame (output)
PART_2 = description of particle created by operation (output)
ISEED = random-number generator seed (input/output)

RECORD /PARTICLE/ PART, PART_2
REAL*8 PARAMS(0:*), EXDIST
INTEGER*4 ISEED
CHARACTER NAME*40
LOGICAL TRANSMIT, FLAG

C********** TYPE _ 7 5 **********

SUBROUTINE TYPE_75(PART, EXDIST, PARAMS, GEOM, ELEMENT, IREG, &
JSURF, KSURF, NAME, TRANSMIT, FIELDS, FLAG, &
PART_2, DET_WT, IX, IY, ISEED)

Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'
INCLUDE 'mc_elmnt.inc'
INCLUDE 'constant.inc'
Variables in calling sequence:
- **PART** = record containing description of particle (input/output)
- **EXDIST** = distance to exit surface particle is aimed at (m) (input/output)
- **PARAMS** = array with description of what is in this region (input)
- **GEOM** = structure with all surface and region definitions (input)
- **ELEMENT** = structure with region space and magnetic parameters (input)
- **IREG** = region number of device, or subregion within device (input/output)
- **JSURF** = surface number, if particle is initially on surface (input)
- **KSURF** = surface number that particle is pointed toward (input/output)
- **NAME** = name of region, used as file name for type 34 (input)
- **TRANSMIT** = flag to compute transmission of sample types 30-39 (input)
- **FIELDS** = flag to include magnetic fields (input)
- **FLAG** = flag set to .FALSE. if (e.g.) chopper in wrong frame (output)
- **PART_2** = description of particle created by operation (output)
- **DET_WT** = statistical weight of detected particle (output)
- **IX, IY** = position bin numbers of detected particle (output)
- **ISEED** = random-number generator seed (input/output)
- **PARAMS(0:*), EXDIST, DET_WT**
- **INTEGER IREG, JSURF, KSURF, IX, IY, ISEED**
- **CHARACTER NAME*40**
- **LOGICAL TRANSMIT, FIELDS, FLAG**

---

**vec_prod**

```fortran
SUBROUTINE vec_prod(cx, cy, cz, ax, ay, az, bx, by, bz)

! Compute vector product (cx, cy, cz) = (ax, ay, az) x (bx, by, bz)
!
! Support function for McStas; original code not found.
!
! Variables in calling sequence
! cx, cy, cz = cross product vector (output)
! ax, ay, az = first input vector (input)
! bx, by, bz = second input vector (input)
IMPLICIT NONE
REAL*8 ax, ay, az, bx, by, bz, cx, cy, cz
!
! No Externals
--
```

---

**WHICHR**

```fortran
LOGICAL FUNCTION WHICHR(PART, GEOM, ELEMENT, IREG)

C Finds which region the particle PART is in (or is entering) by testing
C all boundaries of all regions defined in GEOM (excluding magnetic
C regions). On return, WHICHR = .TRUE. and IREG = region number if successful,
C IREG = 0 if not in any defined region, and if in more than one region
C (neither of which is type 50) IREG is the negative of the lower number
C found. If IREG < 0 at input, only regions > |IREG| are tested.
C
C M.W. Johnson and C. Stephanou, Rutherford Laboratory report RL-78-090.
C 03 Feb 1994: converted from COMMON to STRUCTUREs; added PART and GEOM
C to calling sequence [PAS]
C 11 Apr 1999: IREG = #- instead of -1; check IREG<0 on input [PAS]
```

---
Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'

Variables in calling sequence:
PART = record with particle coordinates (input)
GEOM = structure with definitions of all surfaces and regions (input)
ELEMENT = structure with region parameters (input)
IREG = number of region that /MC_PART/ is in (input/output)
RECORD /PARTICLE/   PART
RECORD /MC_GEOM/    GEOM
RECORD /MC_ELEMENT/ ELEMENT

Externals:
TESTIN
LOGICAL TESTIN

********** W O B B L E **********

SUBROUTINE WOBBLE(PART, SURF, AP, BP, CP, COSTH, ISEED)

Compute the cosine of the angle of incidence of the particle PART with
respect to surface SURF. A vector of length SURF.BETA with random
orientation is added to the surface normal vector, and the resulting
(renormalized) effective surface vector is returned as (AP, BP, CP).
The cosine is returned in COSTH. Downslopes greater than the slope
of the neutron velocity are not allowed; another random slope is
tried. If SURF.BETA<0, a completely random surface is assumed,
independent of ISURF. No test is made to assure that particle is
actually on the surface.

Based on M. W. Johnson, Rutherford and Appleton Laboratories report
RL-80-065.
03 Feb 1994: converted from COMMON to STRUCTUREs; changed calling
sequence [PAS]
20 Sep 1998: change calling sequence to REAL*8, delete BETA [PAS]
31 Oct 2006: exclude slopes that are self-shadowing [PAS]

Definitions of STRUCTUREs:
IMPLICIT NONE
INCLUDE 'mc_geom.inc'

Variables in calling sequence:
PART = record with particle coordinates (input)
SURF = record with surface definition parameters (input)
AP,BP,CP = direction cosines of effective surface normal (output)
COSTH = cosine of angle between particle and surface normal (output)
ISEED = random-number generator seed (input/output)
RECORD /PARTICLE/   PART
RECORD /SURFACE/ SURF
REAL*8   AP, BP, CP, COSTH
INTEGER*4 ISEED

Externals
REAL*8 FUNCTION XCHOPPER(T, TOPEN, TCLOSE, SIGT, V, PERIOD, &
                         LOPENING, ISEED)

Compute the location of the leading or trailing edge (whichever is closer)
 of a mechanical chopper at time T, if the half-open/half closed times are
 TOPEN and TCLOSE, the linear velocity is V (positive if clockwise,
 negative if counter-clockwise), and the random phase jitter is SIGT.
 If LOPENING is returned as .TRUE., the result is the location of the
 opening edge of the chopper; if .FALSE., the closing edge.

20 Sep 1998: change all REAL*4 to REAL*8 (except PLNORM) [PAS]
21 Oct 2002: change MOD() to DMOD() for F90 standard [PAS]

Variables in calling sequence:
T = time that particle reaches chopper (us) (input)
TOPEN = nominal time that chopper is half open (us) (input)
TCLOSE = nominal time that chopper is half closed (us) (input)
SIGT = standard deviation of phase jitter of chopper (us) (input)
V = linear velocity of chopper (m/us) (input)
PERIOD = time between chopper openings (us) (input)
LOPPENING = .true. if chopper is opening, .false. if closing (output)
ISEED = random number seed (input/output)

IMPLICIT NONE
REAL*8 T, TOPEN, TCLOSE, SIGT, V, PERIOD
LOGICAL LOPENING
INTEGER*4 ISEED

Externals:
PLNORM

REAL*4 PLNORM