Title: **Monte Carlo tool for neutron optics
and neutron scattering instrument design**

Author(s): **L.L. Daemen, P.A. Seeger, R.P. Hjelm,
and T.G. Thelliez**

Submitted to:

<http://lib-www.lanl.gov/cgi-bin/getfile?00796859.pdf>

Monte Carlo tool for neutron optics and neutron scattering instrument design

L.L. Daemen, P.A. Seeger, R.P. Hjelm, and T.G. Thelliez

Los Alamos National Laboratory, Los Alamos, NM 87545

ABSTRACT

Unlike x-ray generators, neutron sources have inherently low brightness, and care must be exerted in the design of neutron scattering instruments and their coupling to the source to ensure optimal use of the beam. We present a general, versatile Monte Carlo tool for the computer simulation of neutron optics and neutron scattering instruments that allows a user to produce computer models of an instrument and study its performance quantitatively. The Neutron Instrument Simulation Package (NISP) implements a wide range of neutron optics models to describe neutron transport (including gravity) and scattering in the elements making up the instrument. The program is freely available on the world-wide web at <http://strider.lansce.lanl.gov/NISP/Welcome.html>

Keywords: Monte Carlo, neutron scattering, neutron optics, simulations

1. INTRODUCTION

Neutron optics is a field that has grown significantly over the past 20 years with experimental and theoretical research devoted to the development of new devices and to a better understanding of the fundamental principles on which these devices are based. The field has much in common with x-ray and light optics, as well as with charged particle optics, but is decidedly distinct and unique in many of its aspects. Arguably, the larger size of optics, x-ray, and particle accelerator communities (compared to the community of neutron scatterers) has resulted in the development of large computer codes (such as SHADOW¹) capable of calculating the performance of large complex optical systems. These codes are used routinely for applications ranging from the design of bench top optical systems to the design of synchrotron beam lines and large accelerator structures. In contrast, codes in use in the neutron scattering community are usually *ad hoc*, very limited in scope, and focussed on a specific application (e.g., guides or bent crystals).

The advent of new neutron sources and a worldwide effort initiated a few years ago to build so-called next-generation accelerator-based spallation sources have spurred renewed interest in the design and optimization of sophisticated neutron scattering diffractometers and spectrometers for use in physics, chemistry, materials science and engineering, and many other scientific disciplines. These sources are largely pulsed sources. The complexity of the moderator phase space (emission time, velocity, spatial, and angular distributions) coupled to the desire to make the best possible use of limited intensity requires the use of design tools that go beyond simple analytical and ray-tracing techniques.

Our purpose in initiating the development of the Neutron Instrument Simulation Package (NISP) was to provide the community of neutron scatterers with a general, versatile tool similar to what is available for optical design in other regimes. The starting point was a library of Fortran subroutines (MCLIB) developed by Dr. Michael Johnson at the Rutherford-Appleton Laboratory². The subroutines performed elementary neutron transport operations, with optical element geometry in the form of cells bounded by quadratic surfaces. The user had to write his own program that would call the subroutines in the library to set up an instrument model and run a Monte Carlo simulation. Having extended the library with routines necessary for spallation sources, in 1993 we rewrote MCLIB using structures³. This allowed the Monte Carlo “engine” to be extracted as a completely general separate program (MC_RUN); however the user still had to write a special program to describe the geometry. In 1995, with the widespread advent of the World Wide Web, we began to build a graphical interface⁴ that allows the user to select optical elements from a list, define their characteristics, location, and orientation in space, and write a file to pass the information to MC_RUN. This web application, MC_Web, is the essential third component of NISP. The code has a large collection of source terms, optical elements, and sample and detector types. All the necessary structures are in place to deal with neutron spin, but this is a feature that has not been implemented yet. The structures in the code also permit transport of photons, but this also has not been exploited.

The entire package was built with portability and flexibility in mind . Users can add their own subroutines to model optical elements that are not in the library, either by modifying existing algorithms or adding their own to the library. The package is freely available⁵ and we welcome comments and requests for additions or changes, and we offer technical support to the extent that we have the time to do so.

2. CODE STRUCTURE

Figure 1 shows the relationship between the main components of NISP:

MCLIB: a library of Fortran subroutines that deal with elementary tasks such as geometry representation, neutron transport, and contains all the models for optical elements. Associated with this library is a set of source files representing more than 25 different types of moderators and a set of scattering kernels for sample simulation, as well as various data files with physical constants for materials.

MC_RUN: a Monte Carlo engine that runs the simulation and produces a series of output file with detector output and general information about the outcome of the simulation. *MC_RUN* takes care of neutron propagation in the instrument and tallies neutrons at all relevant places in the instrument.

MC_Web: A WWW-based application that allows the user to set up the instrument geometry interactively without having to learn any of the data structures of *MCLIB*.

Figure 1 – NISP simulation flow chart. The three components of NISP are *MC_Web*, *MC_RUN*, and *MCLIB*. The circles represent files (input or output).

A typical simulation involves the generation of a computer model of the instrument, or portion of instrument to be simulated. The user is assisted by *MC_Web* in this endeavor. When an optical element is selected from a list, *MC_Web* presents the user with a form to fill out with all the information needed to define the an instance of the element. Element parameters are divided into two groups:

- Extrinsic parameters define the position and orientation of the element in space;
- Intrinsic parameters define element characteristics that are independent of its location in space

Figure 2 shows a typical example of the form for a Fermi Chopper. A button next to each parameter name allows to user to bring up a help message relative to the parameter.

After defining all the necessary elements, the user needs to describe their connectivity. Although NISP can transport neutrons anywhere in space, trying to enter any volume that is not defined explicitly will result in the neutron history being terminated. That is, neutrons are not restricted to flow sequentially from a given element to another in a list, but the space between all elements must be defined. Neutron transport occurs by default in optical elements, but gaps between elements are not defined explicitly. In a “linear” portion of the instrument, such as the incident flight path, connectivity is indicated by an input/output matrix. MC_Web then locates the gaps automatically and suggests the creation of an automatic drift space to assure neutron transport in that region of space. In more complex spatial arrangements one can define a “chamber” in which multiple elements are embedded. This is an important advantage over codes that force the user to specify a unique order of optical elements through which the particle is transported (and over analytical methods), in that stray neutrons following unexpected or roundabout paths are also tracked. Such parasitic neutrons can compromise the performance of an instrument. Over the years we have been surprised more than once by unexpected features in our results related to such neutrons.

The screenshot shows a Netscape browser window titled "Los Alamos Neutron Science Center Instrument Modeling Prototype - Netscape". The address bar shows the URL "http://strider.lansce.lanl.gov/cgi-bin/co2.3". The page header includes "Luke L. Daemen", "Los Alamos National Laboratory", and "MC_Web".

On the left side, there is a sidebar menu with the following items: "Fermi_Chopper (tail-cutting)", "Delete this Element", "Duplicate this Element", "Back to Instrument", "Instruments", "Element Types", "Users", "Others", "Calculator", "What's New", and "Information".

The main content area displays the "Element Type: Fermi_Chopper" and "Element Name: Fermi_Chopper". Below this, a message states: "To edit the parameters value, change the data below and press Update at the end of this form."

The parameters are divided into two sections:

- Extrinsic Parameters:**
 - X: mm
 - Y: mm
 - Z: m
- Intrinsic Parameters:**
 - Rotation Frequency: Hz
 - Curvature (reciprocal radius): /m
 - Nominal time-of-flight from source: μ s
 - Phase jitter: μ s
 - Diameter of rotor: mm
 - Height of rotor: mm
 - High-energy macroscopic cross section: /cm
 - Thermal macroscopic cross section: /cm
 - Width of slit package: mm
 - Height of slit package: mm
 - Number of blades:
 - Thickness of blade: mm
 - High-energy macroscopic cross section: /cm
 - Thermal macroscopic cross section: /cm

At the bottom of the form, there is a button labeled "Update Element: Fermi_Chopper (tail-cutting)".

Figure 2 – A typical form presented by MC_Web to the user to define the parameters of an optical element (in this case, a Fermi chopper). Parameters are divided into extrinsic position parameters and intrinsic parameters describing the device.

MC_Web has many additional features. A particularly useful one allows the user to produce a 3D view of the instrument during construction of the model to verify the positioning and orientation of the optical elements in space. MC_Web writes a Virtual Reality Modeling Language (VRML) file for a dynamic picture of the instrument. The user can manipulate the view and click on the various optical elements to bring up the associated form. The latter can be instantly modified and the results of the modification viewed with any VRML viewer. We advocate use of the free COSMO viewer, but the downloaded VRML file may be manipulated with any virtual reality viewer. Figure 3 shows an example of the type of picture produced.

Figure 3 – A 3D view of the HERMES -I backscattering spectrometer proposed for the Manuel Lujan, Jr. Neutron Scattering Center at Los Alamos National Laboratory as part of an ongoing upgrade project. Elements include the source (in the far distance), a chopper, various guides (including curved and converging), a sample (red), two banks of analyzer crystals on a spherical surface, two conical detector banks, and a shielding block between the detectors. The virtual reality view can be manipulated with the controls at the bottom of the screen.

The final result of the user's interaction with MC_Web is the "geometry file" that contains all the relevant instrument information for the simulation. The file is ASCII, with a well-defined structure. It can be edited manually and can even be produced directly without help from MC_Web. This however would require a deep understanding of the data structures of MC_RUN and MCLIB, which was the reason MC_Web was written in the first place.

The geometry file serves as an input file for the Monte Carlo engine, MC_RUN. Figure 4 summarizes the internal structure of MC_RUN. Following initialization, there is a loop to sample the source. A second loop transports neutrons between regions and tests for surface interactions. In the innermost loop, neutrons that penetrate into an optical element are operated on according to an algorithm from the MCLIB library, until they reach a boundary of the region. At that point a neutron can

get lost (if there is no well-defined region of space behind the surface); it can interact again with the same or another element; it can also undergo an interaction at the surface (see below, surfaces can be absorbing, can have roughness, etc).

Figure 4 – The internal structure of the Monte Carlo engine, MC_RUN, showing the main steps in neutron transport through an instrument. There are three nested loops: source, transport between regions, and operations within regions.

Neutrons can be created while interacting within an optical element or at a surface. When this occurs the neutron statistical weight is split (a common variance reduction trick in analog Monte Carlo). For instance, when the neutron reaches a sample, it is automatically split into a transmitted neutron and a scattered neutron. The weights are calculated according to an algorithm specific to each sample type. This guarantees that each time a neutron hits the sample, a scattering event occurs even if the sample scattering cross section is low. In general, splitting is also used to deal with very low probability events. Neutrons produced during the course of a propagation history in the instrument are stored in a bank for subsequent transport and the main neutron continues to be tracked until it is lost, absorbed, or detected. Neutrons in the bank are then transported before another source neutron is started. At the beginning of the simulation, the user specifies the number of histories contributing detected neutrons. The simulation stops when this number of histories is detected or when the number of histories started exceeds 1024 times this number, whichever occurs first.

Another important feature of MC_RUN is its ability to monitor neutrons crossing any surface in the problem and record the position and (true) velocity of the particle as well as weight and spin. This information is not available experimentally on a per neutron basis and is extremely useful to understand the instrument better. The monitor file can also be used as source neutrons for further runs, especially useful for parametric studies.

The output produced by MC_RUN is a pair of ASCII files; the first has 1D, 2D, or 3D histograms for each detector, and the second contains information about the course followed by the simulation, error messages, and various summaries. The histograms produced by MC_RUN can have up to two spatial dimensions and one time dimension (time of flight). The output data file is in ASCII format with a fixed structure. It can be read by the user and manipulated with spreadsheets and plotting packages, or read by a user-written program for further manipulation.

Both MC_Web and MC_RUN make use of the element definitions in the MCLIB library. This library is the heart of the code and is essentially hidden to the general user. Some understanding of the nature of the MCLIB library and its associated data structures is helpful in terms of using the code in general and in deciding on how to set up the instrument geometry. The basic concept in MCLIB is that of a region, which is a portion of space bounded by quadratic surfaces to which a certain

functionality is assigned. A quadratic surface divides space into two regions, and a region is defined by specifying on which side of a surface the region is and by taking the intersection of all these half-spaces. The functionality is embodied by the corresponding MCLIB subroutine (or set of subroutines in complex cases), Figure 5. For instance, a cylindrical piece of material is bounded by a cylindrical surface plus two parallel planes. These surfaces define a region of space (element geometry). The portion of space is filled with a material characterized by a complex scattering length and an absorption cross section that define the functionality of this optical element for neutron transport. This functionality defines a *type*. MCLIB includes many types grouped into classes (see Figure 6).

The separation of geometry and type is convenient in that it allows one to reuse subroutines: *e.g.*, the subroutine responsible for scattering by a crystal can be used to define the type of a region that is either a flat crystal –a parallelepiped- or a curved, spherical crystal. Once the subroutine is written the second element is added to MC_Web with minimum effort, and other crystal shapes could be subsequently added with equal ease.

Figure 5 – The structures of a region in MCLIB.

The basic unit in MC_Web is a pre-defined *element*, created by association of several regions, each of which is characterized by its geometry and type. For instance, a “Pipe” consists of two regions:

- the interior: may be void (vacuum) or filled with a material such as air;
- the wall: may be a perfect absorber or a structural material such as iron or aluminum.

An instance of an element is created in MC_Web when the user specifies a list of parameters associated with the element. These parameters are either extrinsic (they define the position and orientation of the element in space) or intrinsic (they define characteristics inherent to the element), *cf.* Figure 2. Element characteristics and position can be modified at will. Elements can be duplicated and deleted.

An *instrument* is created by positioning elements in space and by defining their characteristics. Typically an instrument has at least a source and a detector. The user can create, modify, delete, and copy entire instruments. The concepts of instrument, element, and region are summarized in Figure 7, and Figure 8 illustrates the generation of an instance of an element.

A *particle* is represented by a record including the particle position, velocity, average spin components, charge, mass, time-of-flight, and statistical weight. The inclusion of charge and mass means that the structures of the code are not limited to just neutrons or photons. If needed, one could consider adding subroutines to handle photons and also charged particles such as electrons. The spin of the particle is a vector representing an average polarization from a statistical spin ensemble.

MC_RUN assigns a value to these parameters upon sampling a source file. The source can be a simple energy spectrum representing a continuous (reactor) source, or a more complex entity including the time and energy dependence of the pulse emitted by a pulsed source. Sources are normally files containing the parameters of an analytical parameterization of the

Type 00-09: Materials (Absorber, Al, hydrogenous, supermirror, Be, filters ...)

Type 10-19: Collimators (multi-aperture, Soller horizontal or vertical, monochromators, mirrors, ...)

Type 20-29: Choppers (X or Y blade chopper; single blade, counter-rotating; Fermi chopper, drum chopper, gravity focuser, ...)

Type 30-39: Samples (constant Q, hard spheres, isotropic elastic and inelastic, reflectometry, powder, ...)

Type 40-49: Detectors (plane, curved; monitors, PSD; 1D, 2D, 3D; Cartesian, cylindrical, or spherical coordinates encoding)

Type 50: Scattering chamber (void) enclosing multiple elements

Type 70-79: Available for *ad hoc* and developmental use

Type 90-99: Sources (time, energy, and spatial characteristics; pulsed or continuous)

Figure 6 – In MCLIB, types are grouped in classes. A type is associated with the functionality of a region. Sub-types of a given type can also be defined (e.g., choppers moving horizontally or vertically are two subtypes of the chopper type).

Figure 7 – Hierarchical representation of the various concepts introduced in the text.

Figure 8. Using extrinsic and intrinsic parameters to generate an instance "Primary Pipe" from the element-definition template, "Pipe."

pulse (from calculated or experimental data), but they can also be a files of neutrons listing the characteristics of each neutron. Such a file can be produced by saving the particle records of neutrons crossing a given surface during a simulation. This monitor file can then be used as a source in subsequent simulations. The source could also be an *ad hoc* file created externally by the user.

3. EXAMPLE

Since the index of refraction for neutrons in most materials is slightly less than unity, there is a critical angle for total reflection at surfaces in air or vacuum. This makes it possible to use reflective optics at glancing angles. Since the difference of the index of refraction from unity is essentially proportional to wavelength, such optics are especially useful for long wavelength neutrons. Figure 9 shows the configuration of a small-angle scattering instrument using an ellipsoidal mirror to form an image of the collimating aperture placed at the first focus onto the detector placed at the second focus. The resolution of the scattering angle when a sample is placed in the beam depends on the size of the beam spot.

Figure 9. Small-angle scattering instrument with an ellipsoidal mirror. Vertical dimensions are exaggerated for clarity. The surfaces of the aperture and the detector are at the foci of the ellipsoid, which are about 2 mm from the vertices.

The mirror is the central portion (4.000 m long by 92 mm wide) of a very eccentric ellipsoid ($a = 10.20$ m and $b = 160.2$ mm), inclined upward by 0.9° . Note that the mirror reflects vertically upward; this corrects to first order for gravitational droop. Otherwise the effect of gravity on the beam spot is significant. Inclusion of gravity is automatic in NISP.

Toroidal mirrors are sometimes used to approximate the ellipsoid . They are less expensive because they are ground in cylinders representing the minor axis, and then deformed to emulate the major axis curvature. We wished to use NISP to compare the optical quality of toroids to ellipsoids. It is very easy to do an ellipsoidal mirror because the ellipsoid is a quadratic surface; however it is not possible to define a toroidal surface as a region boundary. The solution is to define a region type for “Toroidal Surface.” Then the definition of the toroid is by parameter s in the type definition. The region is defined as two materials separated by a toroidal surface. Neutrons are transported within the region, reflecting as often as necessary from the surface, until they reach an outer boundary of the region. Two subroutines were added to MCLIB; one finds the distance to an intersection with the torus, and the other finds the angle of incidence.

Figure 10 (a) The image of a 2 mm diameter aperture produced by an elliptical mirror. Notice the distortion of the image produced by gravity acting on the slow neutrons; (b) An attempt to emulate the effect of the elliptical mirror with a cheaper, easier to fabricate toroid; (c) Improving on the single toroid mirror with eight toroidal segments. Notice the presence of low probability events (blue spots) outside the main image; these are the result of imperfect matching at the boundaries between segments. This background is detrimental to the performance of the small angle diffractometer for which this mirror was designed and cannot be eliminated by collimation.

The results of three simulations are shown in Figure 10. The ellipsoidal mirror behaves well, with the largest apparent defocusing effect being gravity. The toroid image is not satisfactory. Another instrument was simulated with the toroid divided into 8 segments, each with different curvatures to be tangent to the ellipsoid at its center. This produced a tolerable image, but there is evidence of background scatter from the discontinuities of the surfaces at segment boundaries.. Runs were also made (not shown) to test the effect of imperfections in the elliptical surfaces. Errors of 1% in the major and minor axes, and a reasonable amount of surface irregularities produced visible effects in the resolution, but the effects were still small compared to actual detector resolution.

4. CONCLUSION

The complexity of the pulse emitted by a moderator at a pulse neutron source together with the desire for best utilization of the neutrons led us to develop a user -friendly Monte Carlo code to simulate neutron optics and neutron scattering instruments.

The current interface facilitates the interaction between the user and the code to set up and run a simulation. Although a methodology is in place to add new optical elements to the code, it still requires a deep knowledge of the data structures involved as well as of the code itself. A new interface is being developed that will facilitate the insertion of new optical elements by the general users.

New elements are also being added and more sophisticated algorithms are being introduced for existing elements. Since 1998, we have added a Fermi chopper, curved crystal, curved neutron guide, multi -level inelastic scattering sample, and a more general crystal model where the lattice plane are not necessarily parallel to the crystal faces.

In performing this work, we have tried to be responsive to the needs of the neutron scattering community, and our work can go forward only with input and feedback from the community. NISP is freely available, and we encourage users to send comments, criticisms, and requests for additional features. Much code has been written over the years to simulate very specific optical elements and neutron scattering processes. On several occasions we have incorporated such legacy code in NISP, to the mutual benefit of the author and other users of the code.

ACKNOWLEDGEMENTS

We thank many of our colleagues at Los Alamos (Mike Fitzsimmons, Joyce Roberts, Glen Olah, Greg Smith), at Brookhaven National Laboratory (Larry Passell), at the National Institute of Standards and Technology (Uli Wildgruber, John Copley), and at the Rutherford-Appleton Laboratory's ISIS facility (Stuart Campbell, Ron Smith, Mike Johnson), who contributed in one way or another to the development of NISP. We have also benefited from countless discussions with many other colleagues too numerous to mention here; we are no less grateful for their help, support, and insights in making this project a success. This work has been supported by the Los Alamos National Laboratory and the Manuel Lujan Jr. Neutron Scattering Center, 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.

REFERENCES

1. C. Welna *et al.*, "SHADOW: a synchrotron-radiation and x-ray optics simulation tool," *Nucl. Instr. Meth.* **A347**, 344-347, 1994.
2. M. W. Johnson and C. Stephanou, "MCLIB: A Library of Monte Carlo Subroutines for Neutron Scattering Problems," RAL Report RL-78-090; M.W. Johnson, "MCGUIDE: A Thermal Neutron Guide Simulation Program," RAL Report RL-80-065.
3. P. A. Seeger, "The MCLIB Library: Monte Carlo Simulation of Neutron Scattering Instruments," Proceedings of the Thirteenth Meeting of the International Collaboration on Advanced Neutron Sources (ICANS XIII) held at the Paul Scherrer Institut October 11-14, 1995, G.S. Bauer and R. Bercher, Eds., PSI-Proceedings 95-01, p. 194
4. T. G. Thelliez, L. L. Daemen, P. A. Seeger, and R. P. Hjelm, Jr., "A user-friendly geometry interface for the Monte Carlo neutron optics code MCLIB," *ibid.*, p. 307.
5. Current documentation and the source codes are available by anonymous ftp from <ftp://strider.lansce.lanl.gov/NISP/>. Access to the application MC_Web is at <http://strider.lansce.lanl.gov/NISP/Welcome.html>.