LA-UR-04-5645

Approved for public release;
distribution is unlimited.

Title:	THE NEUTRON INSTRUMENT SIMULATION PACKAGE, NISP
Author(s):	Daemen, Luc L. LANSCE-12 Seeger, Philip LANSCE-12
Submitted to:	Annual SPIE Meeting Denver, Colorado USA 08/08/2004

Los Alamos
NATIONAL LABORATORY

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the University of California for the U.S. Department of Energy under contract W-7405-ENG-36. By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

Form 836 (8/00)

Proceedings SPIE **5536** (2004) xxx-xxx

The Neutron Instrument Simulation Package, NISP

Philip A. Seeger* and Luke L. Daemen
Manuel Lujan Jr. Neutron Scattering Center, Los Alamos National Laboratory,
Los Alamos, NM 87545-1663, USA

ABSTRACT

The Neutron Instrument Simulation Package (NISP) performs complete source-to-detector simulations of neutron instruments, including neutrons that do not follow the expected path. The original user interface (MC_Web) is a web-based application, http://strider.lansce.lanl.gov/NISP/Welcome.html. This report describes in detail the newer stand-alone Windows version, NISP_Win. Instruments are assembled from menu-selected elements, including neutron sources, collimation and transport elements, samples, analyzers, and detectors. Magnetic field regions may also be specified for the propagation of polarized neutrons including spin precession. Either interface writes a geometry file that is used as input to the Monte Carlo engine (MC_Run) in the user's computer. Both the interface and the engine rely on a subroutine library, MCLIB. The package is completely open source. New features include capillary optics, temperature dependence of Al and Be, revised source files for ISIS, and visualization of neutron trajectories at run time. Also, a single-crystal sample type has been successfully imported from McStas (with more generalized geometry), demonstrating the capability of including algorithms from other sources, and NISP_Win may render the instrument in a virtual reality file. Results are shown for two instruments under development.

Keywords: NISP, Monte Carlo, simulation, neutron instruments

1. HISTORICAL DEVELOPMENT OF NISP

The relationship of the parts of the Neutron Instrument Simulation Package (NISP) is shown in Fig. 1 (specifically for the Windows version). The oldest element is **MCLIB**, a library of subroutines first written by Mike Johnson in 1978[1]. The philosophy established at that time was to divide all space into *Regions* bounded by quadratic *Surfaces*, and to treat interactions within any region independently from the geometry. This philosophy closely follows the microscopic transport code MCNP (then known as MCN)[2]. The main difference from MCNP is that the algorithms in MCLIB treat neutron *optics* and collective effects; that is, the wave nature of the neutrons is emphasized instead of the particle nature. The library continues to grow, and now includes many kinds of scattering samples, collimation and analyzer devices, detectors, and precession in magnetic fields generated by current loops and solenoids[3,4].

Originally the library was used to support Monte Carlo codes written by individual users. A major development in 1994 was to write a separate Monte Carlo "engine," **MC_Run**, which reads a set of structures from an instrument geometry file[5,3]. The tasks of tallying detector histograms and reporting statistics were standardized, but the user still had the responsibility of creating the geometry file. MC_Run output also can include a monitor file of neutrons crossing a specified surface. The monitor file

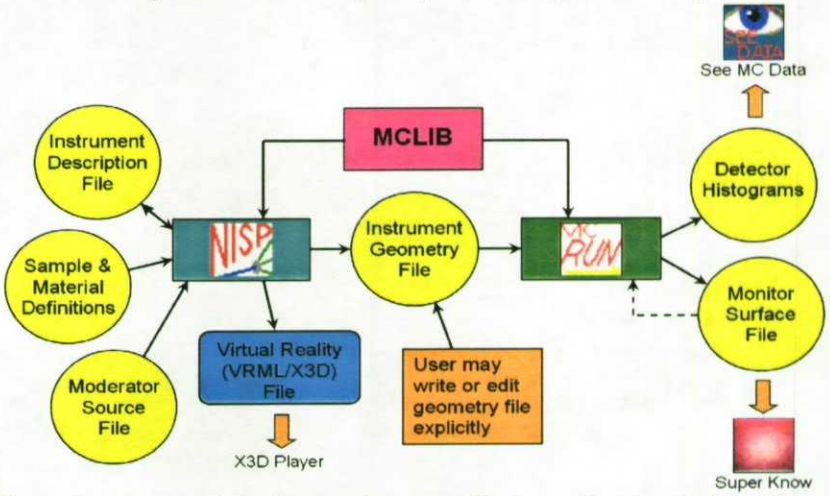

Fig. 1. Components of the Neutron Instrument Simulation Package (NISP). A user interface (either NISP_Win as shown, or MC_Web) writes an Instrument Geometry File with all Surfaces, Regions, and Parameters. The Monte Carlo engine MC_Run performs the simulation, using algorithms from the subroutine library MCLIB.

* PASeeger@aol.com

can be used as the neutron source for a subsequent run.

The final piece (and the name "NISP") was opened to the public in 1995[6]. **MC_Web** is a web application that includes a graphical user interface as well as many database functions such as an instrument library and ability to send instruments to other users. The user builds the instrument on a Los Alamos server[7] by creating instances of *Elements* selected from a menu. Elements consist of one or more regions, the surfaces bounding the regions, and the algorithms that apply within the regions. The geometry file is generated and downloaded to the user's computer, where MC_Run is executed. The instrument can also be rendered in "virtual reality" by a VRML plug-in such as Cosmo[8] in the user's web browser. Further details of the internal structure of the package were presented at a previous SPIE conference[9].

For several reasons, an alternative user interface **NISP_Win** has been written for Windows 98/NT operating systems[4,10]. The reasons include lack of funding to provide updates of MC_Web, difficult or very slow internet access to the server, and no ability to make local modifications or additions. While lacking in the database functions of MC_Web, NISP_Win has added an Instrument Description file that can be archived or shared locally. The newest version (2.0 or later[10]) can also generate a virtual reality file to be viewed by a local X3D player (such as Octaga[11]). As with the rest of NISP, the code is open-source and extensively commented so that it may be used as a template for writing a more general interface. The download includes all source codes (Fortran90), executables, and data files. There are two additional programs for visualization of the resulting histograms (See_MC_Data, specific to Windows) or for extraction of information from a monitor surface file (Super_Know, portable across systems).

2. THE NISP_Win USER INTERFACE

2.1. Code structure

NISP_Win was designed in the Microsoft Visual Studio and compiled as a QuickWin application in Fortran90 (Digital Visual Fortran version 6.0). As a result it contains many arcane references to the Windows operating system, and some Fortran peculiarities. The following description of the functionality is provided for two purposes: to assist in conversion to another system, and to show how additional elements can be included. The flowchart of NISP_Win is shown in Fig. 2. We do not show code details here, but invite reading of the downloaded source codes.

All user input is through dialog boxes. In particular, files are opened using the familiar browsing window. Since this function does not have an interface in the QuickWin environment, it required extra effort to call full Windows APIs (see `OpenNISPFile.f`). Another functionality that requires APIs outside QuickWin is the use of customized Help information, which will be discussed below. Other than these two functions, QuickWin provides all the calls needed for

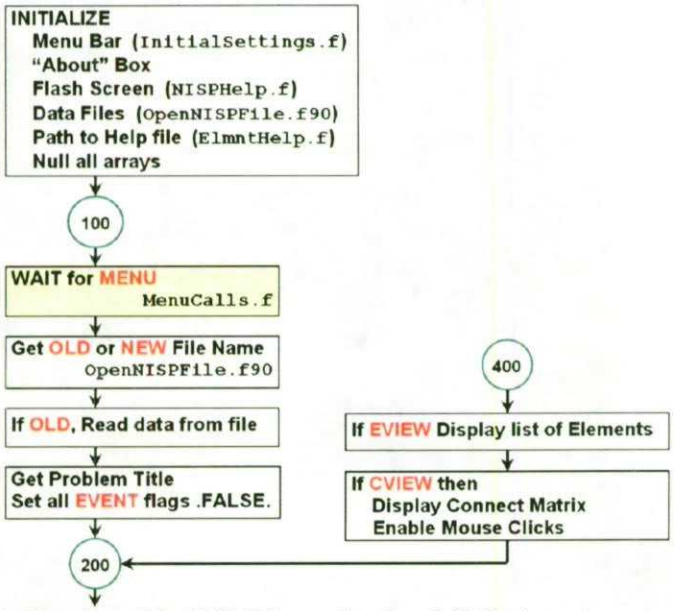

Fig. 2A. Flowchart of the NISP_Win user interface, initialization and repeat section. There are wait loops at 100 and 200. After interrupts are processed, the code loops back at 400.

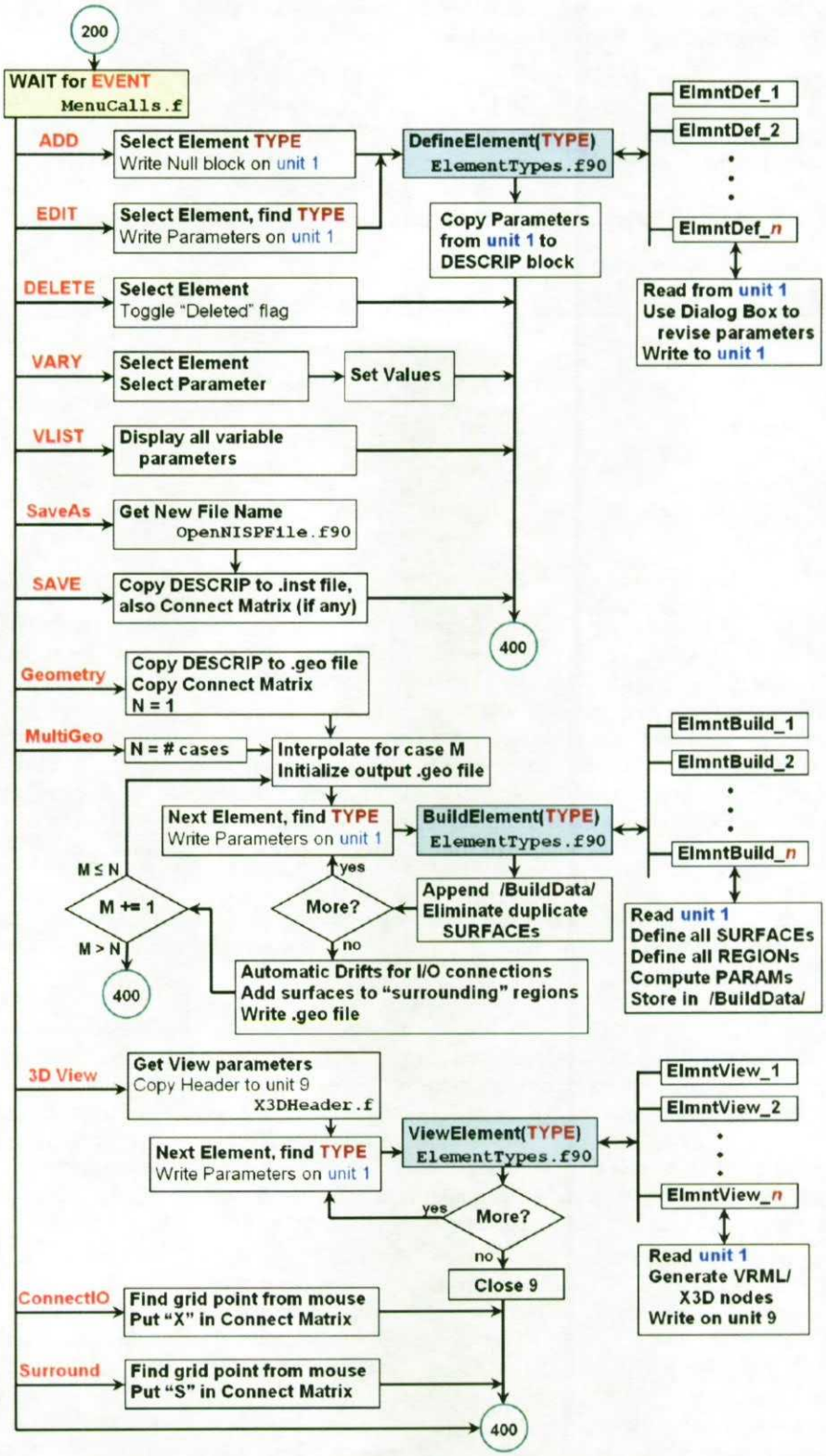

Fig. 2B. NISP_Win flowchart, interrupt processing sections. Functions DefineElement, BuildElement, and ViewElement use CASE structures to choose the method for each specific Element type.

using dialog boxes, the menu bar, and mouse clicks. Visual Studio provides a Resource Editor for the design of dialog boxes, with a full set of controls available for drag-and-drop into the box.

The program NISP_Win is interrupt driven, with wait loops at two locations (100 and 200 in Fig. 2). Most functions are activated through the Menu Bar, as illustrated in Fig. 3, and all of the callback routines except Help are in module `MenuCalls.f`. The callback routines generally set a specific logical variable, but for "Element>View Elements" or "Connect>Enable" events, the flag and a checkmark in the menu are toggled. Two of the event flags represent mouse clicks in the Connect window; for these the position of the mouse is also returned. For "File>Exit" the program is terminated; otherwise control is relinquished, allowing the main program to break out of its loop and test for which event flag was set.

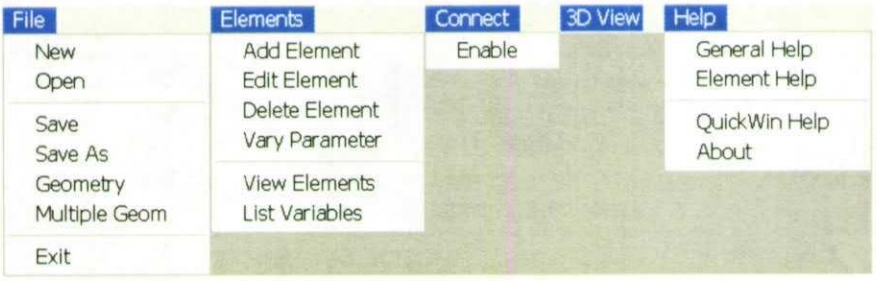

Fig. 3. NISP_Win menu bar and drop-down menus.

The code currently has 37 Element types defined. Since Fortran is not object oriented, internal references are by an integer index in the list of types in `ElementTypes.f90`. Each Element must support three operations and have a set of help pages. As indicated in Fig. 2B, when functions DefineElement, BuildElement, or ViewElement are invoked with a type number, module `ElementTypes.f90` calls the appropriate method. For any given Element, the three operations are entries in a single procedure, to assure consistency of variable definitions. Information is always passed to the procedure by writing a block from an internal copy of the instrument description file onto unit 1. (NAMELIST format is used for generality and in Fortran NAMELIST can only be used with an external file.) The form of information returned is different for each operation. For DefineElement it is a revised NAMELIST, and so is overwritten onto unit 1. For BuildElement it is a partial geometry file with fixed format, which is placed in a common block. For ViewElement a block of X3D nodes is written directly to unit 9.

It should be noted that only the DefineElement functions include system-dependent code, because they rely on dialog boxes for user interaction. The BuildElement and ViewElement functions are intended to be completely portable.

2.2. Example: the "Pipe" element

We shall create an instance of a "Pipe" Element to illustrate the structure of the code. The relevant code is found in the download at \NISP\ElmntDef\ElmntDef_4.for (corresponding to index number 4 for a "Pipe" element). This procedure is a useful template for other types that have the same extrinsic (position and orientation) parameters: X, Y, Z, Tilt in the horizontal plane, and Slope above the horizontal plane. For the example, we place the center of the entrance face of the pipe at the origin, tilt 10° to the left and slope 15° upward. The pipe itself is described by intrinsic parameters. The length is 2 m, inner radius 150 mm, and wall thickness 25 mm. The interior is 'void' and the wall is 'Aluminum' (at 300 K). The Instrument Description File, as seen in the first box in Fig. 4, has the name of the specific instance, the name of the Element object, version information, and the parameters in two NAMELIST blocks.

The ElmntDef_4 procedure displays the appropriate Dialog Box (see Fig. 4), loads it with the current parameter values, and waits for "OK" or "Cancel." All of the parameters are defined in the procedure according to classes which include real, integer, character string, real array (20), yes/no check boxes, and multiple choice radio buttons. Character strings are further divided into material or crystal identifiers, or file names for source or powder data. All data necessary to draw the dialog box, including the static text descriptions, are included in the procedure, although in the Windows environment the boxes are predefined and stored in a resource file (`Script1.rc`). The token for every dialog box is 100 + the type index, and the tokens for the edit and list boxes in every dialog box begin with 1000 for the element name and are sequential; radio buttons begin at 1100 and check boxes at 1150 (see file `resource.fd`). Following a standard allows the code that loads and reads the parameters to be common across all element types.

Instrument Description File

```
Pipe w/tilt & slope [1]
Pipe
Version 1.2, 05 Mar 2000, P.A.Seeger
 &EXTRINSIC
 X        =  0.000000000000000E+000,
 Y        =  0.000000000000000E+000,
 Z        =  0.000000000000000E+000,
 TILT     =   10.0000000000000      ,
 SLOPE    =   15.0000000000000
 /
 &INTRINSIC
 LENGTH   =   2.00000000000000     ,
 RADIUS   =   150.000000000000     ,
 WALL     =   25.0000000000000     ,
 ROUGHNESS     =  0.000000000000000E+000,
 MATERIAL1     = 'void       ',
 MATERIAL2     = 'Aluminum   '
 /
```

Dialog Box

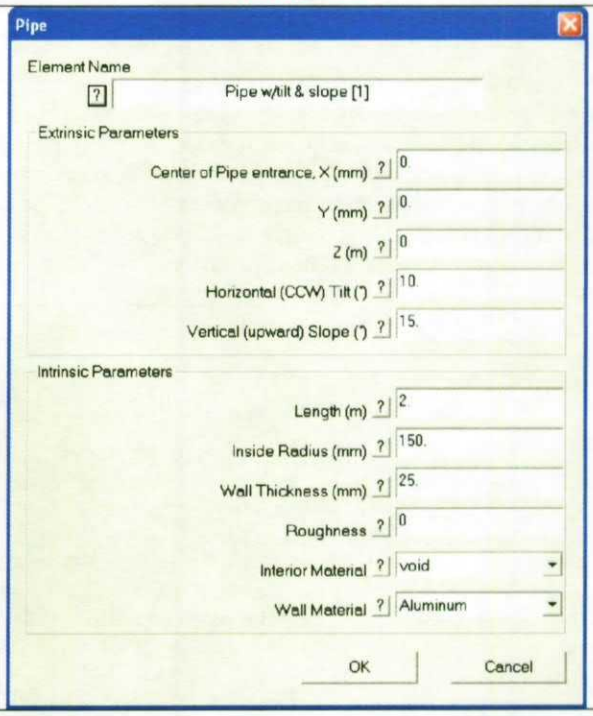

Fig. 4. Description file block for "Pipe," and corresponding Dialog Box generated by DefineElement (ElmntDef_4).

The ElmntBuild_4 function uses the parameters from the Instrument Description file to compute the data for the geometry file (shown in Fig. 5). For instance, all surfaces must be expressed in world coordinates in the form[1]

$$A x^2 + B x + C y^2 + D y + E z^2 + F z + G + P x y + Q y z + R z x = 0. \quad (1)$$

The pipe has four surfaces: entrance plane, interior and exterior cylinders, and exit plane. All coefficients depend on the Tilt and Slope angles. Regions are defined as being either on the + or the – side of the surfaces by whether eq. (1) evaluates to be > 0 or < 0. The interior of the pipe, for instance, is on the + side of the first plane, the – side of the inner cylinder, not bounded by the outer cylinder, and on the – side of the exit plane. The region definitions form a matrix with rows for regions and columns for surfaces. The function must also provide a token for each surface to show how it

Partial Geometry File in /BuildData/

```
 4 (Surfaces)   2 (Regions)   2 (Parameters)
 0  -.1677312594965  0  .2588190451025  0  .951251242564  0  0  0  0  0      (1st Surface)
 .971866224588  0  .933012701892  0  .0951210735201  0  -.0225  .0868240888335
     -.492403876506  .3191091380258  0
 .971866224588  0  .933012701892  0  .0951210735201  0  -.030625  .0868240888335
     -.492403876506  .3191091380258  0
 0  -.1677312594965  0  .2588190451025  0  .951251242564 -2  0  0  0  0
-INPUT, NONE, +EXTERIOR, +OUTPUT  (Connection flag tokens for the 4 Surfaces)
   1  -1   0  -1     (Surface relationships of 1st Region)
   1   1  -1  -1
'Pipe w/tilt & slope [1] interior          '
'Pipe w/tilt & slope [1] wall              '
    0    1           (Parameter pointers for the 2 Regions)
 2.1  300            (Definition of Aluminum at 300K)
```

Fig. 5. Geometry file block for "Pipe," plus flags for connectivity to other elements, output from BuildElement (ElmntBuild_4).

may connect to the rest of the world. The record in Fig. 5 between the surface and region definitions shows these tokens: Input to the element is from the – side of the first surface, and Output is to the + side of the last surface. The outer cylinder surface is Exterior on its + side. Parameters in this example are simple. A pointer value of 0 indicates 'void', and aluminum has a region code of 2.1 and a single parameter, T = 300K. This is all that MC_Run needs.

The ElmntView_4 function converts the shape parameters from the Instrument Definition File into a third representation of the geometry, shown in Fig. 6. The origin shift and the Tilt/Slope angles are accomplished simply using two transform nodes, and the intrinsic description of the pipe is a Shape node[12,13]. Note that the signs of Z-coordinates generally have to be changed, because the VRML coordinate system is always right-handed while NISP uses left-handed coordinates. The shape of a Pipe is generated by defining a long skinny rectangle which is a cross section through the wall, and "extruding" that rectangle in a circle around the Z-axis. (In manufacturing terms, the pipe is rolled and welded rather than being extruded.) The appearance "PipeColor" is defined as blue in the header of the output file.

VRML / X3D File

```
<!-- Pipe w/tilt & slope [1]                              -->
    <Transform translation="0 0 0"                    (Origin)
               rotation="0 1 0 .1745329">              (Tilt)
      <Transform rotation="1 0 0 .2094395" >(Slope)
        <Shape>
          <Appearance USE="PipeColor" />
          <Extrusion crossSection="0.15 0
              0.175 0  0.175 -2  0.15 -2  0.15 0"
                     spine="0 0 0  0 0 0  0 0 0
                       (total of 49 triplets)    0 0 0"
                     orientation="0 0 1 0
                       0 0 1 .1309  0 0 1 .2618
                       (49 rotations from 0 to 2π)
                       0 0 1 6.152  0 0 1 6.283"
                     creaseAngle="0.5"
                     solid="false"
                     beginCap="false"
                     endCap="false" />
        </Shape>
      </Transform>
    </Transform>
```

View with X3D Player

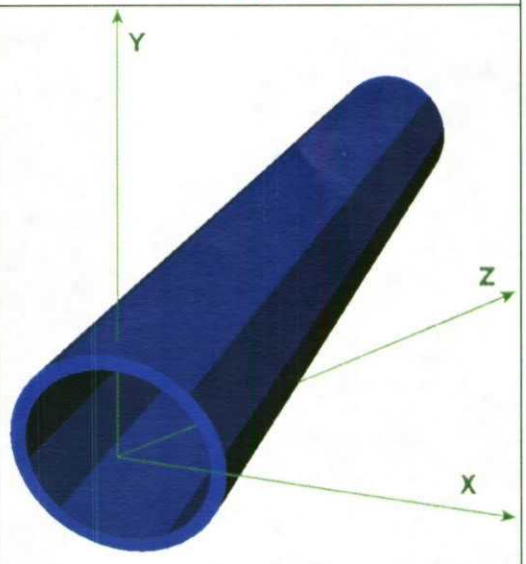

Fig. 6. Virtual reality file block for "Pipe" generated by ViewElement (ElmntView_4), and a rendition by an X3D player.

2.3. Connecting and building a geometry file

The unique feature of NISP compared to other available packages is that *all* space is defined and neutrons may proceed in *any direction* through whatever elements they encounter. This allows simulation of multiple detector banks, and often leads to the discovery of paths contributing to backgrounds. There are three ways to connect elements. First, any elements with a common surface are always connected, in either direction. Second, the user may specify that the output of one element is allowed to go to the input of another element. Third, a group of elements can be specified as being surrounded by another element, as for instance within a scattering chamber, which then interconnects the entire group. Fig. 7 is an example of a connection matrix. A letter "**X**" indicates that the output surface of the row element is to be connected to the input surface of the column element, and a letter "**S**" shows that the row is surrounded by the column. The actual connections must be made in the process of writing a geometry file.

As seen in Fig. 2B, when a geometry file (or multiple geometry files, if variables have been defined) is requested, the program loops through all Elements and calls BuildElement for each. As each block of Surfaces, Regions, and Parameters is returned, NISP_Win tests for duplicate surfaces. Only the original Surface is kept, and the entries in the new Region matrix are modified accordingly. Thus connection across identical surfaces is assured. After all Elements have been built, the connect matrix is checked for **X**s. If the two surfaces to be connected are *not* identical, a "drift" region is automatically defined between them. Finally, **S**s in the connect matrix are processed by flagging all exterior surfaces of embedded regions as being *virtual* surfaces in the surrounding region. When a neutron is traveling in a

chamber or other surrounding region, intersections with every such virtual surface are tested to see if there is a valid Region on the other side.

2.4. Context-sensitive Help pages

Every user entry in the dialog box of every Element has a pushbutton for Help (*cf.* Fig. 4). Clicking the "?" next to the Element Name box calls a page with a description of the Element and links to additional pages. (From the menu bar, Help>Element Help shows a table of contents with links to all Element pages.) Following the links or clicking the "?" of a particular datum will bring up the appropriate page. This is an *essential* feature of NISP, but generating a file for the Windows help system is difficult. We will outline the steps here, and call attention to the files in the \NISP\Help\ folder for use as templates.

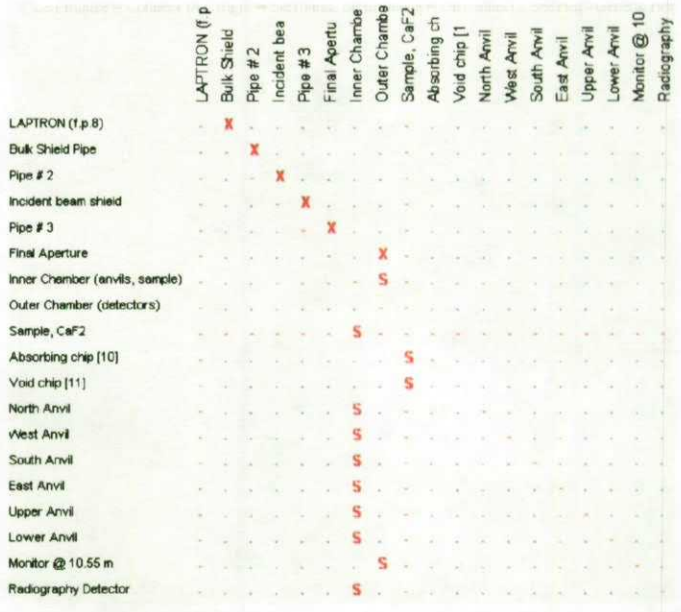

Fig. 7. Connection matrix. "**X**" indicates connection from row to column, and "**S**" means the row is surrounded by the column. There are another 14 detectors surrounded by the "Outer Chamber."

It is possible to write HTML pages for a new Element using WordPad (which doesn't know about HTML) to edit an existing file from \NISP\Help\html\. Another option is to use a simple HTML editor, such as "Scott's WebWriter,"[14] but to maintain style it is still good to use a template. Note that a single anchor point can be used for a group of related variables with a single description for the group. To cause an anchor to begin a new page, precede it in the HTML file with this action (to be converted to an RTF tag):

```
<HTMLTOHLP action="raw-rtf"> {\page} </HTMLTOHELP>
```

Microsoft characters such as °, μ, and Å may be used, but Greek letters and others from the Symbol font require special action. For example, to enter "Δλ", type the following:

```
<HTMLTOHLP action="raw-rtf"> {\f3\fs18 Dl} </HTMLTOHELP>
```

Drawings to be included should be saved in \NISP\Help\bmp\, preferably as bitmapped images.

The second step is to convert from HTML to a special form of RTF using the shareware utility HtmlToHlp[15]. The first file should be \NISP\Help\html\Help0.html, and the output should be directed to \NISP\Help\NISP. To allow for editing of the RTF files, turn off the option to run the WinHelp compiler directly from HtmlToHlp; there is a check box in the "Conversion..." option. The RTF files are sequentially numbered corresponding the Element index number, in folder \NISP\Help\rtf\. Any pages that used the Symbol font as described above will have to be edited to include the definition of "\f3." Use *only* Notepad to do this editing; any higher level editor will destroy unique (non-standard) features of the RTF file, and it won't work. Find the definition of \f2 near the beginning of the file, and following it insert

```
\f3\ftech Symbol;¤
```

(where ¤ represents the <NUL> character). There are presently 12 files that require this font (0, 1, 6, 10, 11, 13, 14, 16, 17, 24, 25, and 30). You should use WordPad to proofread the files, but do *not* Save!

One file generated by the HtmlToHlp utility is a "Help Project file," \NISP\Help\NISP.hpj. Opening this file will launch the Microsoft Help Workshop application. Generally, you only need to push the "Save and Compile" button in the lower right corner. The final step is to look at the file \NISP\Help\NISP_map.h and make sure that the TopicTag for the starting page of each element agrees with the list in module `ElementTypes.f90`. For instance, the line

```
#define PAGE_xx nnn /* Default for this page */
```

means that the TopicTag for Element #*xx* is *nnn*. All other help entries are derived from this in the callback routine ElmntHelp.f by knowing that the dialog box tokens for the Help buttons are sequential.

3. NEW FEATURES

New features are being added continually, as requested by users and as time allows. Most changes and updates are listed in the "What's New" button on the opening page of the web site.[7] We are actively soliciting open-source algorithms that can be included in the library. In particular, the ability to track spin precession in magnetic induction fields[16] (either uniform, interpolated, or generated by loops and solenoids) has not yet been exploited by having routines for various polarization-dependent devices. We list here the significant additions to the package since the last published report.[4]

3.1. Additions to MCLIB

The functions that compute the attenuation length of Al and Be now have temperature as a parameter, with defaults 300K for Al and 100K for Be. The form and parameters used are from Freund[17], with an additional approximation for the temperature dependence of the thermal diffuse term for wavelengths longer than the Bragg cutoff. We treat the $1/v$ cross section as

$$\sigma(\lambda,T) = \lambda \left[\sigma_{abs}(1\text{Å}) + C_3 (T/\Theta_D)^{3/2}\right], \tag{2}$$

where σ_{abs} is the absorption cross section, Θ_D is the Debye temperature, and the arbitrary constant C_3 is 0.102 b/Å for Al and 6.55 b/Å for Be. The temperature dependence to the power 3/2 is an empirical approximation.

A neutron traversing a magnetic field gradient will experience a small acceleration due to interaction with its dipole moment. To allow for future implementation of such devices as a magnetic sextupole lens, routines have been added for distance computation (`DIST_A.f`) and motion (`MOVE_A.f`) including a constant acceleration vector. The treatment is the same as for gravitational acceleration, assuming only a second-order correction to the trajectory, but applied to all three axes.

Both neutrons and x-rays are transmitted through smooth glass capillary tubes by multiple grazing-angle reflections from the interior surface[18]. In both cases the index of refraction in the glass wall is less than unity, leading to total external reflection below a critical angle. A routine (`CAPILLARY.f`) has been added to MCLIB to compute transmission through a tube of fixed radius, bent at a constant radius of curvature. The wall may be described either by the critical angle, or by the complex scattering-length density. In the latter case the complex Fresnel reflection function is used, showing the effect of absorption at the surface. (One result of the simulations is that lead glass, and *not* a borosilicate glass, must be used for neutron optics.) The function solves the exact quartic equation for a torus (with surface roughness), and is based on the existing toroidal mirror function. Comparison to experimental data[19] using optics manufactured by X-Ray Optical Systems (XOS) is very good, as shown in Fig. 8. The algorithm used by XOS is a local quadratic approximation instead of a full quartic, but it has the advantage of being able to vary the diameter and curvature of the capillary[20]. Since no advantage is seen with the quartic, a new version is being written with a "centripetal acceleration" term which should be essentially the same as the (proprietary) XOS algorithm.

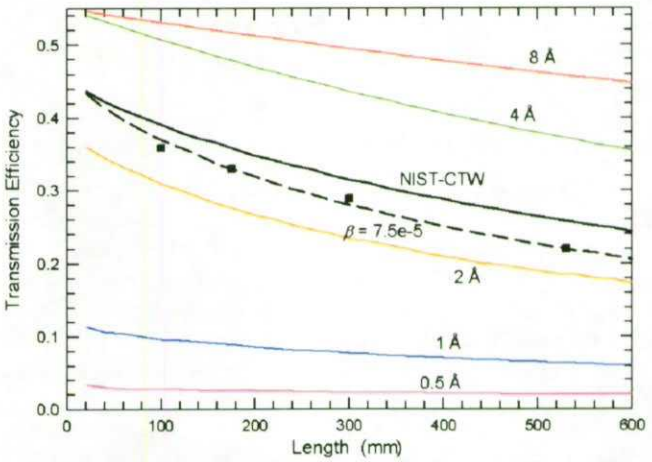

Fig. 8. Transmission through straight fibers of various lengths for several neutron wavelengths, compared to experimental data[18]. The heavier line is simulated using the spectrum at the CTW station of the NIST reactor, where the measurements were made. The dashed line shows the improved fit to the data when a small surface waviness is included.

The single-crystal component from the McStas code[21] has been ported into MCLIB as a sample type (`SNGLXTAL.f`). This is the first time that an external algorithm has been incorporated and we are very grateful to author Kristian Nielsen for his help. The transfer involved translation from C++ to Fortran90, mostly by global editing, followed by conversion of pointer arrays to simple structures. Then all references to geometric shape and neutron transport were removed, as the geometry and content of regions are completely separate in NISP, automatically providing more geometric generality.

3.2. Additions to MC_Run

Run-time options in MC_Run are activated by setting values in a NAMELIST at the start of execution. Three new options have been added. Two of these are simple flags: `GRAVITY='N'` will cause the neutron mass to be set to 0 so the acceleration of gravity is turned off; `DUMPS='Y'` will cause intermediate dumps to be saved in separate files instead of each overwriting the previous one. (The dumps occur at one hour of elapsed time and every two hours thereafter.)

The foremost new option, activated by `TRACE='Y'`, is to trace individual neutron trajectories using the free portable PGPlot library[22] to generate a graphics window. The mouse and keyboard are used to adjust the location, scale, and transverse magnification of the plot, and to switch between plan and elevation views. Various keystrokes show between 1 and 10,000 trajectories, or turn the trace off. The plot shows only trajectories, and not any components or surfaces. Color coding is that incident trajectories are green, splitting into yellow (scattered) and blue (transmitted) when the neutron hits a sample, and turning red for a "bad" neutron (such as coming through a chopper out-of-phase, or reflecting an even number of times in an analyzer crystal). A sample trace centered on the sample of the LAPTRON instrument (described below) is shown in Fig. 9.

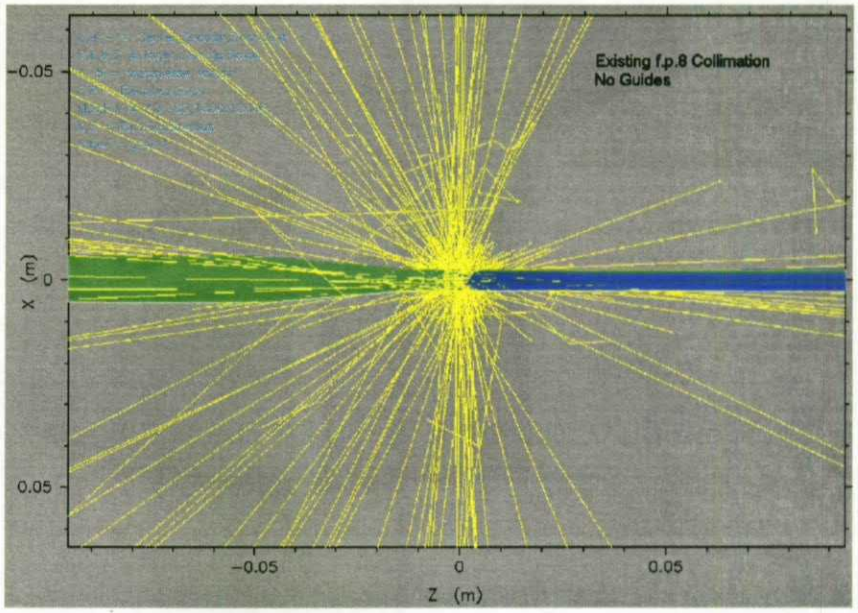

Fig. 9. Trajectory trace (plan view) at the sample of the LAPTRON instrument. See Fig. 11 for a perspective view of the instrument. Although surfaces are not shown, they can be inferred from the behavior of the neutrons.

MC_Run now loops back to the beginning of the program instead of terminating at the end of a job. Batch jobs are to be run by placing all the input data sequentially in a single file. Certain functions (`BREGION`, `GRAV_FOC`, `KERNEL`, and `SNGLXTAL`) are reinitialized before the loopback. The default values in the NAMELIST will be the same as were set by the previous job, except that the default for the random number seed will be the final previous random number.

3.3. Additions to NISP_Win

A new "Single Crystal Sample" element type has been added to NISP_Win, to take advantage of the subroutine imported from McStas. This is the first element in NISP that is *not* axially symmetric, so it was necessary finally to define the third Euler angle as an extrinsic parameter. We define orientation as 1) *Tilt*, rotation about the Y-axis; 2) *Slope*, rotation about the tilted X-axis; 3) *Rotate*, rotation about the tilted and sloped Z-axis. This is illustrated in Fig. 10.

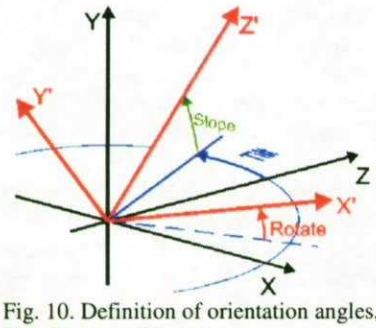

Fig. 10. Definition of orientation angles, Tilt, Slope, and Rotate.

New source files have been generated for the four moderators in ISIS target 2, and also for the latest version of target 1 (Ta has been replaced by W). These tables are also available in MC_Web.

The most significant addition to the NISP (for Windows) package is the virtual-reality rendition of the instrument. This required a ViewElement entry to be added to every element definition, to generate a block of VRML code[12] (in X3D syntax[13]) such as the example for "Pipe" in Fig. 6 above. This expands and in some cases improves the rendition already available in the web-based interface. The .x3d file can be edited easily: to delete an element, change the string "-->" at the end of the first comment line so that the whole element becomes a

comment. To see what is inside an element, change its Appearance to "`USE="VoidColor"`" instead of its natural color; in the example of LAPTRON in Fig. 11, the incident pipe at the left has thus been made translucent to see the guide inside it. The fun part of the rendition is navigating through the instrument.

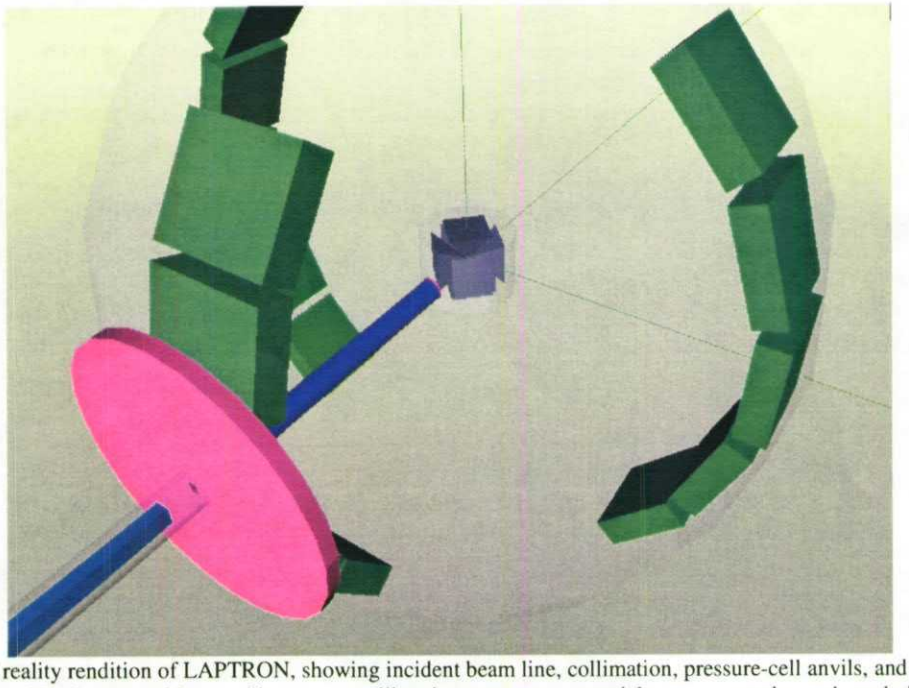

Fig. 11. Virtual reality rendition of LAPTRON, showing incident beam line, collimation, pressure-cell anvils, and detector banks. In the virtual world, pipes are blue, guides cyan, collimation magenta, materials gray, samples red, and detectors green. Navigation tools provided by Octaga[11] include Walk and Fly, Examine and Lookat, Slide and Pan.

4. LAPTRON, ANVIL CELL DIFFRACTOMETER AT LOS ALAMOS

The Los Alamos Neutron Science Center is exploring the possibility of the building a dedicated high-pressure neutron scattering instrument, LAPTRON, to conduct *in-situ* neutron diffraction and neutron radiography & tomography experiments under simultaneous high-pressure and high-temperature conditions.

A newly commissioned 2000 ton press will be permanently installed inside LAPTRON in a stand-up position and will use a newly designed ZIA-type anvil package. The ZIA (*Z- Intrusion Anvils*) module is a modified *d*-DIA anvil package with slightly acute-angled (80°) anvils and a "geared" metal gasket insertion to block the extrusion of the pressure medium. The ZIA module with its cubic cell assembly will provide better hydrostatic pressure and homogeneous temperature conditions, while maintaining anvil gaps for diffraction and radiographic windows. The four vertical gaps between the horizontal anvils, which can be seen in Fig. 11, allow diffraction at right angles ($2\theta = \pm 90°$) and also back-scattering ($2\theta = \pm 135°\text{-}175°$) for higher resolution. Simultaneously, the gap in the forward direction allows neutron radiography using the transmitted neutrons. The cubic cell assembly will also permit ultrasonic elasticity measurements and "drop-flow" viscosity measurements together with high *P-T* diffraction experiments. Following the pioneering high *P-T* deformation work with the *d*-DIA anvil package, the ZIA module can purposely introduce a deformation component for the study of rheological properties at high pressures. This function is particularly beneficial to the *in-situ* high *P-T* studies of yield strength and texture development as a structural phase transition occurs in crystalline materials. This is the type of sample environment that has a definite impact on the instrument performance, but is difficult to model accurately with simple analytical expressions. Monte Carlo methods, on the other hand, can deal with the complexity of the high-pressure sample environment described above without much difficulty. The feature of NISP that neutrons are traced through multiple paths (*cf.* Fig. 9) is essential. Navigation through Fig. 11 demonstrates the paths available to the neutrons. Monte Carlo simulations with NISP show that it is in principle possible to obtain a gain in intensity by a factor of five compared to the current state-of-the-art powder diffractometer (HIPPO) at Los Alamos while preserving the resolution of the HIPPO instrument. Tapered guides increase intensity with a tolerable reduction of resolution. Fig. 12 shows a typical simulated diffraction pattern, for a configuration including the guides.

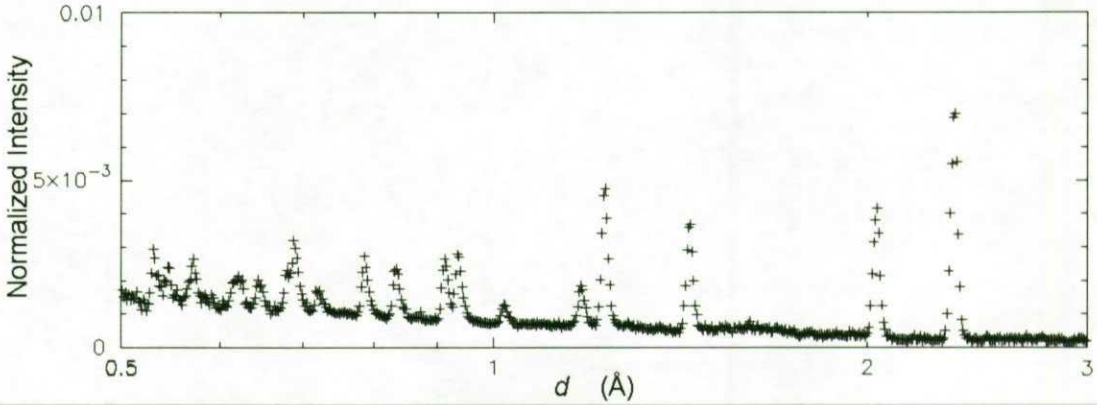

Fig. 12. Aluminum powder diffraction pattern on LAPTRON. Optimization of the instrument with NISP shows that it is possible in principle to design and build a high-pressure instrument (with low sample volume) that could acquire refinable diffraction data for such a simple crystal in seconds, or a more complex sample in minutes.

Simultaneous radiography/tomography and diffraction (at high-*PT*) is becoming increasingly attractive to geologists and material scientists. This type of measurement will be routinely available on LAPTRON. Monte Carlo simulations represent a natural approach to optimizing a radiography setup. We are proposing to use a capillary optic device to bring the image plane for the radiography close to the pressure cell, in the gap between the anvils, as shown schematically in Fig. 13. The sample in the simulation has an inclusion which is a 1-mm cube of pure absorber to test the radiographic resolution; results are shown in Fig. 14. This simulation was made simultaneously with the diffraction data in Fig. 12.

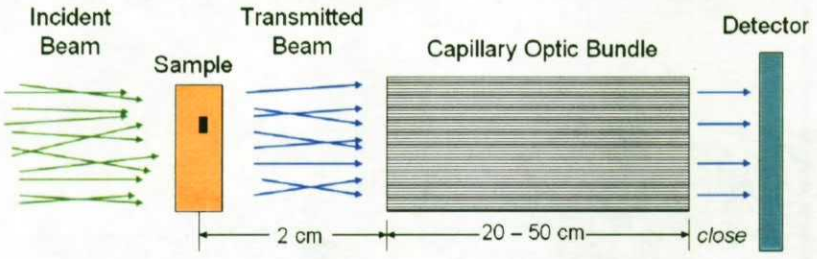

Fig. 13. Use of a capillary optic as a collimator for radiography.

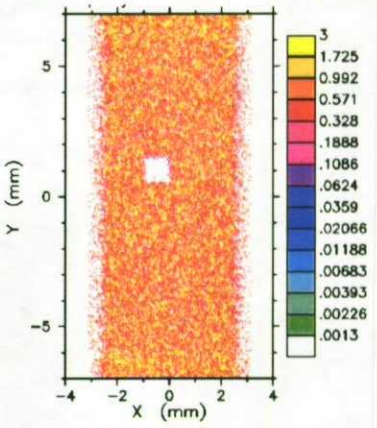

Fig. 14. NISP simulation of a radiography experiment on LAPTRON. The sample has an inclusion of a neutron-opaque 1-mm cube.

5. VISION, CHEMICAL VIBRATION SPECTROMETER AT SNS

One of the next generation instruments for construction at the SNS is VISION, a TOF spectrometer for investigating molecular vibrational dynamics and structure by simultaneous inelastic scattering and diffraction – the neutron analogue of an IR/Raman spectrometer with powder diffraction capabilities[23]. VISION's design criteria include a relatively large dynamic range (10-4000 cm^{-1}), with moderate to high resolution (0.5 to 2 %) across the dynamic range, and data collection rates fast enough to enable kinetic studies and the investigation of non-hydrogen-containing species. These attributes make VISION ideally suited to address a wide range of scientific problems in such fields as biochemistry, catalysis, materials science, geology, polymers and engineering inaccessible today.

Inverted geometry spectrometers define the scattered beam energy, E_2, and scan the incident neutron energy, E_1, by time-of-flight. Operation of the spectrometer is illustrated schematically by the time-distance diagram shown in Fig. 15. The final energy is selected by Bragg reflection of the scattered neutrons on a pyrolytic graphite crystal, as illustrated in

virtual reality in Fig. 16 and trajectory tracing in Fig. 17. The mosaic-crystal algorithm in NISP uses multi-step transport with an analytic solution of the Darwin equations at each step[24]. The design is taken from the TOSCA-II instrument[25] at ISIS, with the addition of the curvature of the analyzer crystal to give focusing in the vertical plane. The NISP simulations indicate that the focusing configuration gives a better signal-to-noise ratio.

The use of a crystal to select the final energy has two distinct advantages:

1. The spectrometer operates with a wide range of energy loss values.
2. In an energy loss experiment, the crystal works at low energy (approximately 3.5 meV for pyrolytic graphite with a Bragg angle of 45°) while time-of-flight is used to determine the higher incident energy.

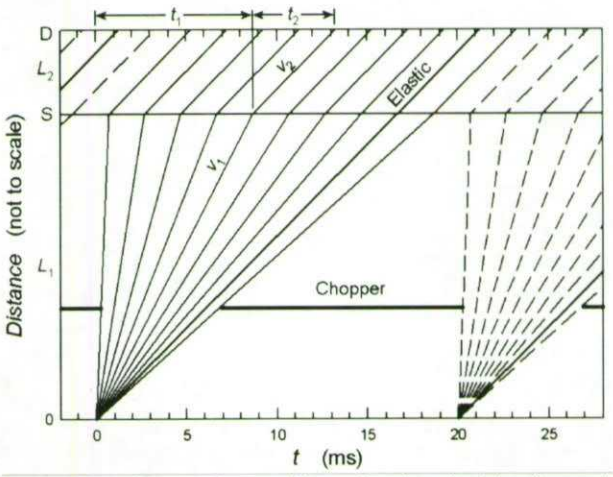

Fig. 15. Time-distance diagram illustrating the principle of operation of a crystal analyzer spectrometer such as VISION. S is the sample location and D the detector. The total flight time of elastic neutrons is accurate, but L_2 is greatly exaggerated (actual t_2 = 0.88 ms). The chopper selects the dynamic range to measure neutron energy loss.

An additional advantage of the use of a crystal is the distinctive simplicity of a passive device as opposed to the use of a mechanical device (such as a chopper) to perform the final energy selection. A proper spatial arrangement of the crystal also permits the capture of a large solid angle of scattered neutrons and thus increased count rates, at little cost in resolution.

One disadvantage of using a crystal for energy selection is that it reflects not only the desired wavelength, λ, but also λ/n, n=2,3,4,... which leads to spurious peaks in the spectrum. The standard solution to this problem is the use of a filter (such as the cooled Be shown in Fig. 16) to eliminate the higher-order reflections.

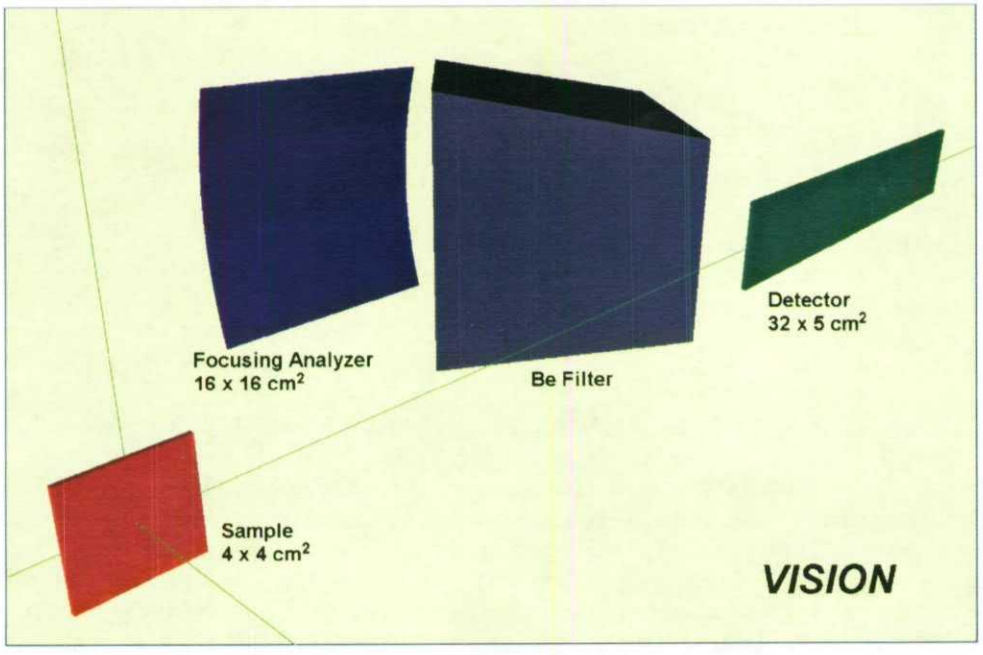

Fig. 16. Crystal analyzer geometry for the VISION instrument. The wavelength of scattered neutrons from the sample is selected by Bragg reflection in the analyzer, and a cooled Be filter is used to remove higher-order contamination. The design is taken from TOSCA-II at ISIS[24] with the addition of focusing. In VISION the analyzer will focus in the vertical direction allowing the detector to be shorter, thereby improving the signal-to-noise ratio.

Fig. 18 illustrates the general agreement between Monte Carlo simulations and experimental results for TFXA, a predecessor of TOSCA-II at the ISIS facility at the Appleton-Rutherford Laboratory. The figure reflects the level of agreement that can be obtained routinely today with a Monte Carlo code such as NISP. However the benchmarking is not complete, as there are now additional data[26] from the two configurations of TOSCA, and the observed pulse shapes in the raw data are not in good agreement with the simulation. In particular, the simulation has not included details of the data acquisition system and the data reduction procedures used. This work continues and will be reported elsewhere.

It is also possible to calculate the resolution analytically, if one takes adequate care in dealing with the combination of *statistically independent* terms: time distribution, sample width, height and thickness, analyzer thickness, and detector element width, height, and thickness. The coefficients of the terms depend on instrument geometry. This exercise provides insight as to which parameters have the largest (or smallest) effect on instrument resolution, and thus guides the design and the Monte Carlo simulations. Complete benchmarking will include experiment, analysis, and simulation..

With the caveat that the benchmarking is *not* complete, we proceed to simulate the intensities and resolution of the proposed VISION instrument, using a "sample" with nine δ-function energy levels between 0 and 400 meV. As seen in Fig. 19, the pulse shapes are asymmetric, resulting from the tails of the moderator pulse shapes. To decrease the effect of the tails in computing the resolution, Gini's mean-difference statistic[27] was used as the estimator of the standard deviation. The probability for each level is the same, but the intensities vary because each energy samples a different part of the incident spectrum. For ΔE less than about 10 meV the average absolute resolution is 0.10 meV, while for higher energies the relative resolution is about 0.9%.

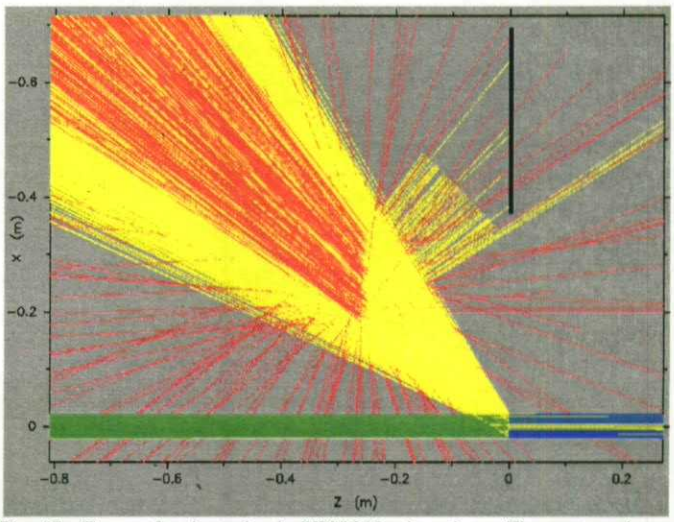

Fig. 17. Trace of trajectories in VISION, plan view. The neutron beam (green) enters from the left and strikes the sample at Z=0 m. The scattering angle has been arbitrarily limited to directions that may hit the analyzer. For 10,000 starts, only 5 "good" neutrons (yellow) reach the detector (which has been drawn in). There are many "bad" neutrons from non-Bragg scatter or even numbers of scatters in the analyzer; these are shown in red, and one bad neutron has been detected. NISP allows an estimate of this source of background.

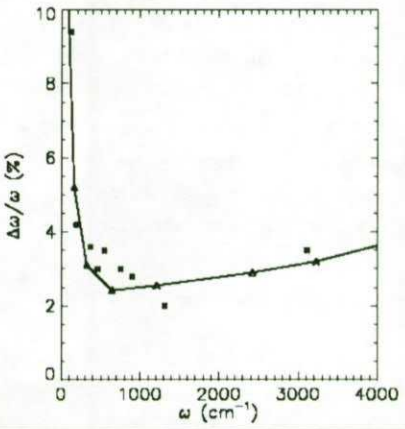

Fig. 18. Measured resolution of the former TFXA (solid squares) spectrometer at ISIS, compared to a Monte Carlo simulations (triangles connected by lines). The excitation energy is expressed in wave numbers, $\omega = \Delta E/hc$, with $1/hc = 8.065$ cm^{-1}/meV.

6. CONCLUSIONS

One decade after NISP was launched as a complete package, it is gratifying to observe that the use of Monte Carlo techniques for neutron scattering instrument design has gained a certain degree of prominence in the community, as exemplified by the proliferation of computer codes and the now-routine requirements by new facilities to provide Monte Carlo verification of instrument performance. Several factors have contributed to this state of affairs. Faster computers and user-friendly software have definitely contributed to making Monte Carlo design tools more accessible to the non-specialist. Sustained benchmarking and code comparison efforts have increased confidence in the validity of the Monte Carlo approach and the reliability of the tools. On the other hand, adequate funding to support the efforts of the software

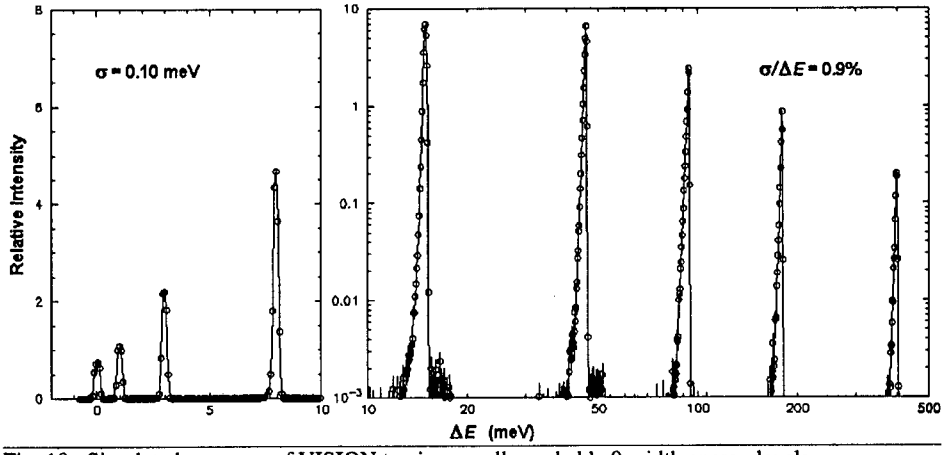

Fig. 19. Simulated response of VISION to nine equally probable 0-width energy levels.

developers remains a critical issue. This is somewhat puzzling because the software development has come at a very modest cost, mainly by adapting tools (an example being VRML) developed for other markets.

Overall the future remains bright. The software is approaching the point where it will be possible to perform simulations in real time to optimize experiments on an actual diffractometer or spectrometer. Virtual reality tools should soon enable a scientist to visualize neutron transport inside a model of an instrument to start addressing background and noise problems that in practice are very difficult to identify and resolve.

In addition to the use of Monte Carlo in increasingly complex and more realistic systems, we also expect to see progress toward the automatic generation of computer code based on the particular needs of the user. Currently the turn-around time between the request for new features and the implementation and testing of these features is relatively long–typically weeks to months. NISP, as described above, has taken a modest step in this direction. But computer scientists are currently actively designing tools for this purpose. Success would provide the ultimate flexible Monte Carlo code for the end-user.

Another foreseeable development is the integration of Monte Carlo codes with other software packages, *e.g.*, for automatic instrument geometry generation or the analysis or conversion of the Monte Carlo data. Seamless integration of Monte Carlo software in more comprehensive software packages would greatly increase productivity and encourage use of the tool by more users.

Finally, let us restate our request for user-supplied algorithms to be included in NISP.

ACKNOWLEDGEMENTS

This work has been funded in part 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. VISION studies were funded by ORNL Program Development funds.

REFERENCES

1. M. W. Johnson and C. Stephanou, "MCLIB: a library of Monte Carlo subroutines for neutron scattering problems", Rutherford Laboratory report RL-78-090, 1978; M. W. Johnson, "MCGUIDE: a thermal neutron guide simulation program", Rutherford and Appleton Laboratories report RL-80-065, 1980.
2. J. F. Briesmeister, ed., "MCNP – a general Monte Carlo n-particle transport code", Los Alamos National Laboratory report LA-12625-M, 1993.
3. P. A. Seeger, L. L. Daemen, R. P. Hjelm, Jr., and T. G. Thelliez, "The neutron instrument Monte Carlo library MCLIB: recent developments", *Proceedings of the 14th meeting of the International Collaboration on Advanced Neutron Sources*, J. M. Carpenter and C. A. Tobin, eds., Argonne National Laboratory report ANL 98/33, vol. 1,

202-218, 1998.

4. P. A. Seeger and L. L. Daemen, "The neutron instrument simulation package NISP: recent developments", *Proceedings of the ICANS-XVI*, G. Mank and H. Conrad, eds., European Spallation Source report ESS 03-136-M1, vol. 1, 483-496, 2003.
5. P. A. Seeger, "The MCLIB library: Monte Carlo simulation of neutron scattering instruments", *Proceedings of the meetings ICANS-XIII and ESS-PM4*, Paul Scherrer Institut Proceedings 95-02, vol. 1, 194-212, 1995.
6. T. G. Thelliez, L. L. Daemen, P. A. Seeger, and R. P. Hjelm, Jr., "A user-friendly, graphical interface for the Monte Carlo neutron optics code MCLIB", *Proceedings of the meetings ICANS-XIII and ESS-PM4*, Paul Scherrer Institut Proceedings 95-02, vol. 1, 307-311, 1995.
7. http://strider.lansce.lanl.gov/NISP/Welcome.html
8. Computer Associates International, http://ca.com/cosmo/home.htm, v2.1 (2000).
9. L. L. Daemen, P. A. Seeger, R. P. Hjelm, and T. G. Thelliez, "Monte Carlo tool for neutron optics and neutron scattering instrument design", *Radiation sources and radiation interactions*, E. J. Morton, ed., *Proc. SPIE* **3771**, 80-89, 1999.
10. ftp://strider.lansce.lanl.gov/pub/NISP/NISPforWindows.zip
11. E. Ottar, http://octaga.com/, v.1.5.0 (July 1, 2004).
12. A. L. Ames, D. R. Nadeau, and J. L. Moreland, *VRML 2.0 Sourcebook, Second Edition*, Wiley, New York, 1997.
13. http://www.web3d.org/x3d/content/X3dTooltips.html
14. S. Hjelmstrand, http://www.webwriter.dk/english/, v3.5.2e (Feb. 3, 2001).
15. F. Allimant, http://www.allimant.org/javadoc/htmltohlpe.html, v0.99h (March 1, 2002).
16. P. A. Seeger and L. L. Daemen, "Numerical solution of Bloch's equation for neutron spin precession", *Nucl. Inst. Meth. A* **457**, 338-346 (2001).
17. A. K. Freund, "Cross sections of materials used as neutron monochromators and filters", *Nucl. Inst. Meth.* **213**, 495-501, 1983.
18. M. A. Kumakhov and V. A. Sharov, "A neutron lens", *Nature* **357**, 390-391, 1992; H. Chen, R. G. Downing, D. F. R. Mildner, W. M. Gibson, M. A. Kumakhov, I. Yu. Ponomerev, and M. V. Gubarev, "Guiding and focusing neutron beams using capillary optics", *Nature* **357**, 391-392, 1992.
19. Q. F. Xiao, H. Chen, D. F. R. Mildner, R. G. Downing, and R. E. Benenson, "A comparison of experiment and simulation for neutron guidance through glass polycapillary fibers", *Rev. Sci. Instrum.* **64**, 3252-3257, 1993.
20. Q. F. Xiao, I. Yu. Pomonerev, A. L. Kolomitsev, and J. C. Kimbal, "Numerical simulations for capillary-based x-ray optics", *X-ray detector physics and applications*, R. B. Hoover, ed., *Proc. SPIE* **1736**, 227-238, 1993.
21. K. Nielsen, http://neutron.risoe.dk/download/components/samples/Single_crystal.html, report Risø-R-1175 (EN), pp. 72-79 (December 1999).
22. T. J. Pearson, ftp://ftp.astro.caltech.edu/pub/pgplot/pgplot5.2.tar.gz, v.5.2.2 (February 26, 2001).
23. J. Z. Larese, B. Hudson, and L. L. Daemen, "VISION, a neutron vibrational spectrometer for SNS", American Conference on Neutron Scattering, College Park, Maryland, June 6-10, 2004, 141 (abstract).
24. P. A. Seeger and L. L. Daemen, "Mosaic crystal algorithm for Mont Carlo simulations", *Appl. Phys. A* **74** (Suppl.), S1458-S1461, 2002.
25. S. F. Parker, C. J. Carlile, T. Pike, J. Tomkinson, R. J. Newport, C. Andreani, F. P. Ricci, F. Sacchetti, M. Zoppi, "TOSCA: a world class inelastic neutron spectrometer", *Physica B* **241-243**, 154-156, 1997.
26. Z. A. Bowden, M. Celli, F.Cilloco, D. Colognesi, R. J. Newport, S. F. Parker, F. P. Rippi, V. Rossi-Albertini, F. Sacchetti, J. Tomkinson, and M. Zoppi, "The TOSCA incoherent inelastic neutron spectrometer: progress and results", *Physica B* **276-278**, 98-99, 2000; http://www.ifac.cnr.it/tosca/risoluz.htm.
27. C. W. Akerlof, "Efficient algorithms for estimating the widths of nearly normal distributions", *Nucl. Inst. Meth.* **211**, 439-445, 1983.