General features of the neutron instrument simulation package VITESS

K. Lieutenant[a]*, G. Zsigmond[b], M. Fromme[a], S. Manoshin[a]

[a]*Hahn-Meitner-Institut Berlin, Glienicker Str. 100, D-14109 Berlin, Germany*
[b]*Paul Scherrer Institut, CH-5232 Villigen PSI, Switzerland*

Abstract

Version 2.5.1 of the software package VITESS that simulates neutron scattering instruments is introduced. The concept of VITESS including data handling and piping is explained. Particular emphasis is given to the use of tools, the options to run several simulations in a series and to split and stop the instrument simulation. General features like ray-tracing and the possibility to stop the simulation without loss of data are described. The role of examples as a starting point for users and for tests of the package is discussed.

Key words:

MC simulation; neutron scattering, instrumentation

***Corresponding author:**

Klaus Lieutenant, Hahn-Meitner-Institut Berlin, Dept. SF1, Glienicker Str. 100, D-14109 Berlin, Germany
email: lieutenant@hmi.de, phone: +49-30-8062-3076, fax: +49-30-8062-3094

1. Introduction

Scientific experiments on neutron scattering instruments generally are more expensive than those on most other instruments. Therefore, sophisticated instruments have been developed that make efficient use of neutrons. This development was strongly supported by the use of Monte Carlo programs which simulate the behaviour of neutrons from the moderator to the detector. Nowadays, every new instrument is simulated intensively before its construction and also changes on existing instruments are first simulated.
During the last years, several packages that simulate neutron scattering instruments have been developed, such as NISP [1], RESTRAX [2], McStas [3], VITESS [4] and Ideas [5]. Their reliability has been proved in a comparison of simulations [6]. Here we present the package VITESS (Virtual Instrumentation Tool for the ESS) with emphasis on questions of software design.

2. History and present status of VITESS

VITESS has been developed since 1998. Version 1 was released in 1999 and version 2.0 containing polarization, absolute flux values and an improved GUI in June 2001. Since September 2004, version 2.5.1 an upgrade of version 2.5 is available. The first modules were written by C. Guy, J. Stride, G. Zsigmond, F. Streffer and D. Wechsler. Now the development

is continued by K. Lieutenant, S. Manoshin and G. Zsigmond. M. Fromme was responsible for the graphical user interface and the releases all the time.
The package was initiated by F. Mezei who had the idea to realize a package that is well suited to simulate instruments on neutron spallation sources. As planned, it was used to perform several simulations for the ESS instrument suite [7-9]. In the meantime it is used all over the world for all kinds of sources and instruments. Until October 2004, more than 370 different people have downloaded the instrument.
VITESS is written in C and can be run on Windows, Unix and Linux computers. It is free of charge. It is easy to handle because it can be used without changing any code or using any script or meta-language - in contrast to other packages. But each user can also write and use his own modules. It has its strengths in a well-developed suite of polarization and time-of-flight modules. VITESS has been supported by the network SCANS (Software for computer aided neutron scattering) in FP5 and is now supported by MCNSI (Monte Carlo simulation of neutron scattering instruments) in FP6.

3. Concept and general features of VITESS

Figure 1 shows the concept of VITESS. Each component in an instrument is represented by one module. The modules are delivered as executables for the various platforms. The executables run independently in a pipe. This piping concept reduces the necessary memory for a simulation. On the other hand it allows fast calculations.
Properties of the component are either transferred as parameters to the module or read from a parameter file. In both cases, the user of the program enters these data via a graphical user interface (GUI, cf. Fig. 4 or 6). This is a Tcl/Tk application (cf. http://www.tcl.tk).
Each module writes position and orientation of (the end of) the component into a file named 'instrument.inf' (see Fig. 2). In this way, information about the instrument as a whole is collected. This will be used for an instrument drawing in the next version.

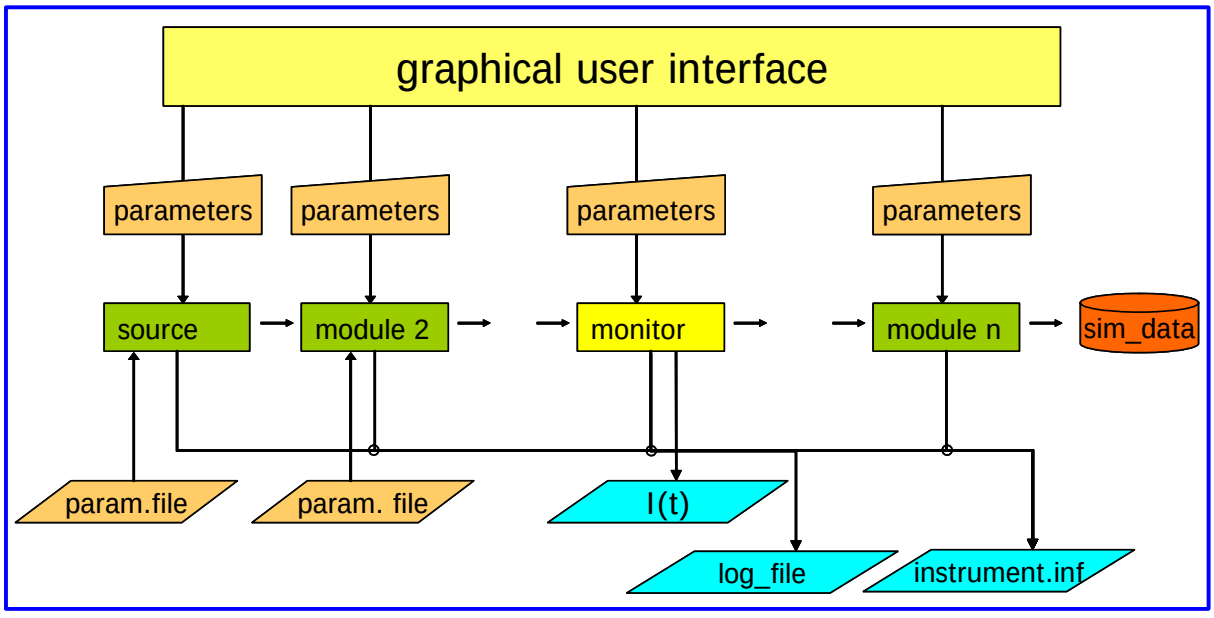
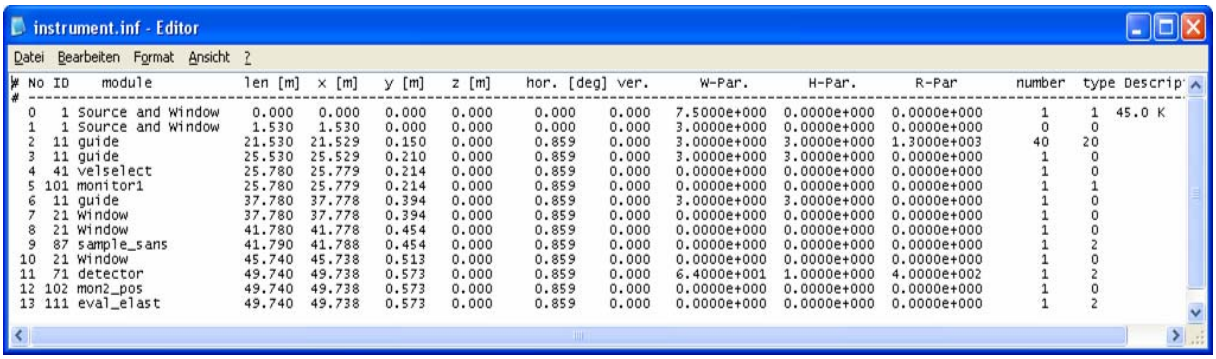


Figure 1: Concept of the VITESS program

The source module creates neutron packages defined by position, divergence, spin, wavelength, (initial) time and count rate sampling the possible trajectories. All independent parameters are defined by Monte Carlo choices within ranges given by the user. This

parameter set changes during the 'flight' through a component. The values at the end of each component are transferred to the following module. In this way, the trajectory through the instrument is calculated. In a mathematical sense, each neutron package is a random event; in VITESS these events are called 'trajectories'.
Some modules write information in a reduced form into a file, e.g. count rate as a function of time, (see Fig. 1). The whole information (complete parameter sets of all trajectories) can either be written into a binary file by the last module ('sim_data' in Fig. 1), or at any point of the instrument into an ASCII file by using the module 'writeout'.


instrument.inf - Editor

Datei Bearbeiten Format Ansicht ?

# No	ID	module	len [m]	x [m]	y [m]	z [m]	hor. [deg]	ver.	W-Par.	H-Par.	R-Par	number	type	Descrip
0	1	Source and Window	0.000	0.000	0.000	0.000	0.000	0.000	7.5000e+000	0.0000e+000	0.0000e+000	1	1	45.0 K
1	1	Source and Window	1.530	1.530	0.000	0.000	0.000	0.000	3.0000e+000	0.0000e+000	0.0000e+000	0	0	
2	11	guide	21.530	21.529	0.150	0.000	0.859	0.000	3.0000e+000	3.0000e+000	1.3000e+003	40	20	
3	11	guide	25.530	25.529	0.210	0.000	0.859	0.000	3.0000e+000	3.0000e+000	0.0000e+000	1	0	
4	41	velselect	25.780	25.779	0.214	0.000	0.859	0.000	0.0000e+000	0.0000e+000	0.0000e+000	1	0	
5	101	monitor1	25.780	25.779	0.214	0.000	0.859	0.000	0.0000e+000	0.0000e+000	0.0000e+000	1	1	
6	11	guide	37.780	37.778	0.394	0.000	0.859	0.000	3.0000e+000	3.0000e+000	0.0000e+000	1	0	
7	21	Window	37.780	37.778	0.394	0.000	0.859	0.000	0.0000e+000	0.0000e+000	0.0000e+000	1	0	
8	21	Window	41.780	41.778	0.454	0.000	0.859	0.000	0.0000e+000	0.0000e+000	0.0000e+000	1	0	
9	87	sample_sans	41.790	41.788	0.454	0.000	0.859	0.000	0.0000e+000	0.0000e+000	0.0000e+000	1	2	
10	21	Window	45.740	45.738	0.513	0.000	0.859	0.000	0.0000e+000	0.0000e+000	0.0000e+000	1	0	
11	71	detector	49.740	49.738	0.573	0.000	0.859	0.000	6.4000e+001	1.0000e+000	4.0000e+002	1	2	
12	102	mon2_pos	49.740	49.738	0.573	0.000	0.859	0.000	0.0000e+000	0.0000e+000	0.0000e+000	1	0	
13	111	eval_elast	49.740	49.738	0.573	0.000	0.859	0.000	0.0000e+000	0.0000e+000	0.0000e+000	1	2	

Figure 2: File 'instrument.inf' that collects information about the whole instrument (Here: SANS instrument at HMI)

4. Survey of modules and examples

4.1 Modules

Version 2.5.1 contains 28 modules representing hardware including 7 sample modules, 7 modules for treating polarization and 5 modules for monochromating components. There are 9 additional modules; they are needed to visualize data: the monitor modules display intensity or polarization as a function of one or two parameters like time, wavelength, vertical position etc. The module 'visual' enables visualisation of trajectories during the run. Two modules ('eval_elast' and 'eval_inelast') allow a first data evaluation, e.g. showing intensity as a function of scattering angle. The module 'frame' allows various kinds of changes of the co-ordinate sysem and thus enables the simulation of geometries which were originally not planned, e.g. a simulation of a bender that is curved in a vertical plane.
Additionally, there is a module called 'external command' which can be used by any user to include modules written by himself. A detailed description of the existing modules is given elsewhere [10-11].

4.2 Examples

Examples are an essential part of the package. 12 different example instruments installed on all kinds of sources are delivered with the package. They can be used by users as a starting point for their own simulation.
On the other hand, these simulations are reference simulations to check the validity and the speed of simulations with new versions of the program. Therefore we have tried to include each module in at least one example, (not fully realized yet.)

5. Tools

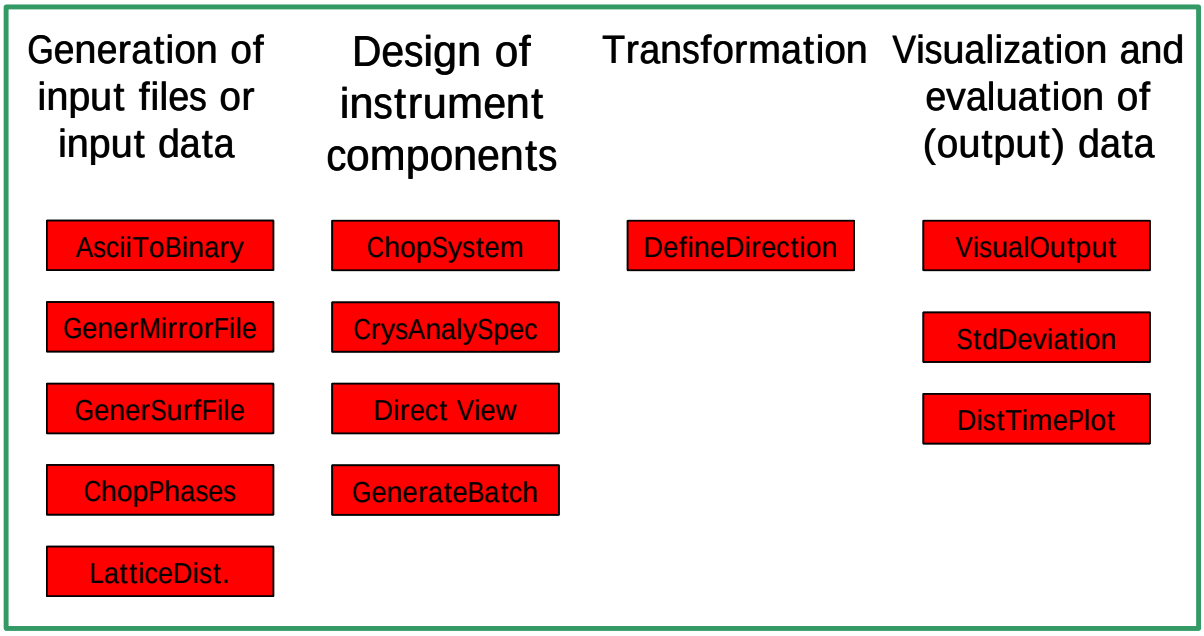


Figure 3: Survey of tools in VITESS

Since version 2.2, VITESS contains tools. They are used to generate input files, to calculate or check input data or to visualize results. The tools existing in version 2.5.1 are displayed in Figure 4. Most of them use a command shell to collect input data (e.g. 'GenerateMirrorFiles', 'AsciiToBinary'). Others operate with Tcl/Tk windows as the main program (e.g. 'DesignChopperSystem', see Figure 4).
'AsciiToBinary' transfers an ASCII output file as it is created by the module 'writeout' into a binary file that can be used as input for the simulation (cf. section 6.1)
'GenerateMirrorFile' creates reflectivity files of coatings used in neutron guides. They are used in the modules 'guide' and 'bender'. 'GenerateSurfaceFiles' generates files that contain information about the geometry of a bender.
'LatticeDistances' creates the input file for a powder-sample from atomic properties, but at the moment only for crystals having FCC lattice.
'ChopPhases' calculates the initial phase of a chopper (in a certain distance to the moderator) needed to let a neutron of a certain wavelength pass. 'DesignChopperSystem' additionally calculates the optimal chopper aperture for a frame overlap chopper and the usable evaluation time (and the corresponding wavelength range) at the detector (see Fig. 4).
'DirectView' checks if a bender is long enough to prevent direct view of neutrons through a system straight guide - bender - straight guide.
'CrystalAnalyzerSpectrometer' calculates the set-up of a crystal analyzer spectrometer [12].
'GenerateBatch' creates a batch file to perform several simulations one after the other. This feature is now realized by the option 'series of simulation' (cf. section 6.3), which is more convenient in its use. Therefore, 'GenerateBatch' will be removed soon.
'DefineDirection' transforms vectors between different co-ordinate system definitions: Cartesian - Spherical - Euler and performs vector rotations.
'VisualOutput' uses a binary or ASCII output file of a VITESS simulation ('sim_data' in Fig. 1) to create several figures: Intensity as a function of wavelength, ….
'StdDeviation' calculates the integral, average value and standard deviation of an intensity distribution, e.g. position, width and intensity of a peak.

'DistTimePlot' creates a figure time-of-flight vs. flight path. This is useful to visualize chopper systems.

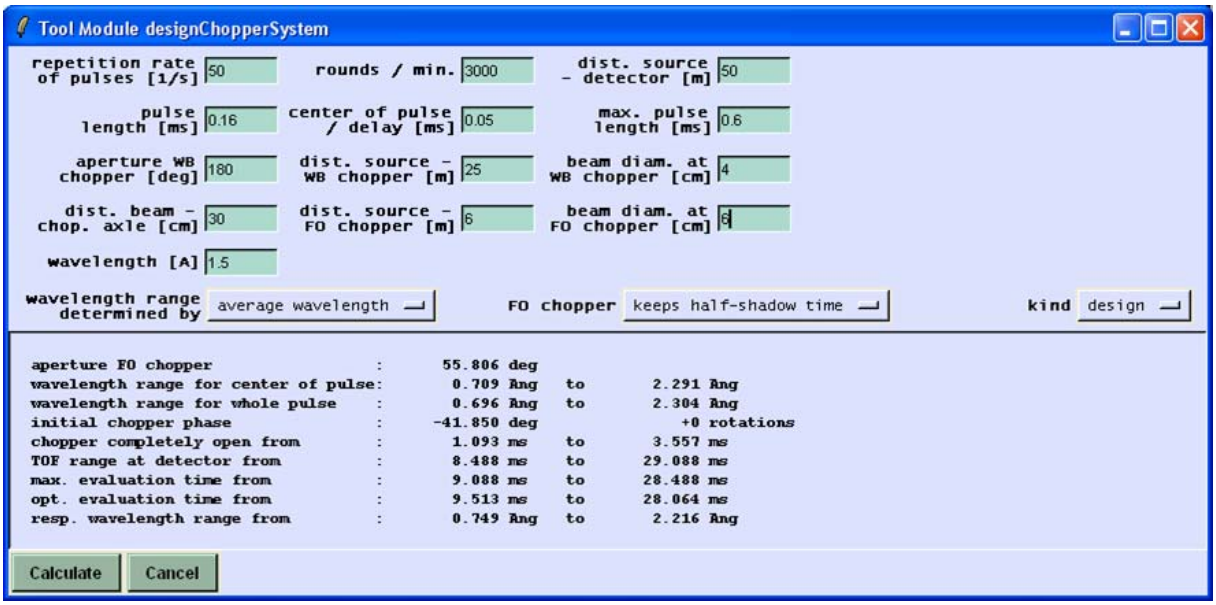


Figure 4: Window of the tool 'DesignChopperSystem'

6. Special features

6.1 Splitting the instrument simulation

As the whole information of all trajectories can be written into a file (cf. section 3), the simulation can easily be split: the last module of the first part writes this information into a file, and the first module of the second part reads it as input. Thus, the file replaces the 'source' module. As input file a binary file is necessary, but this can easily created from an ASCII file by the tool 'AsciiToBinary' (see section 5). As the parameter set of one trajectory has a size of about 0.1 KByte (in ASCII format), this file can be very large. Therefore, it is reasonable to split after several modules, where the number of trajectories is already decreased.
Performing the first part of the instrument only once with a high number of trajectories and then running the second part many times (e.g. for different sample orientations) can save a lot of time.

6.2 Stopping the simulation

The simulation can be stopped without loss of data. The principle is shown in Fig. 5: After each generation of a new trajectory, the source module checks if an interrupt flag has been set. In this case the creation of new trajectories is stopped. But all trajectories already created will be processed by all modules. Therefore a correct simulation (with a lower number of trajectories) is performed.
The only difficulty is that the calculation of absolute flux values is wrong because the contributions of the remaining trajectories are missing. But this can easily be corrected through multiplying by a factor $N_{planned}/N_{started}$.

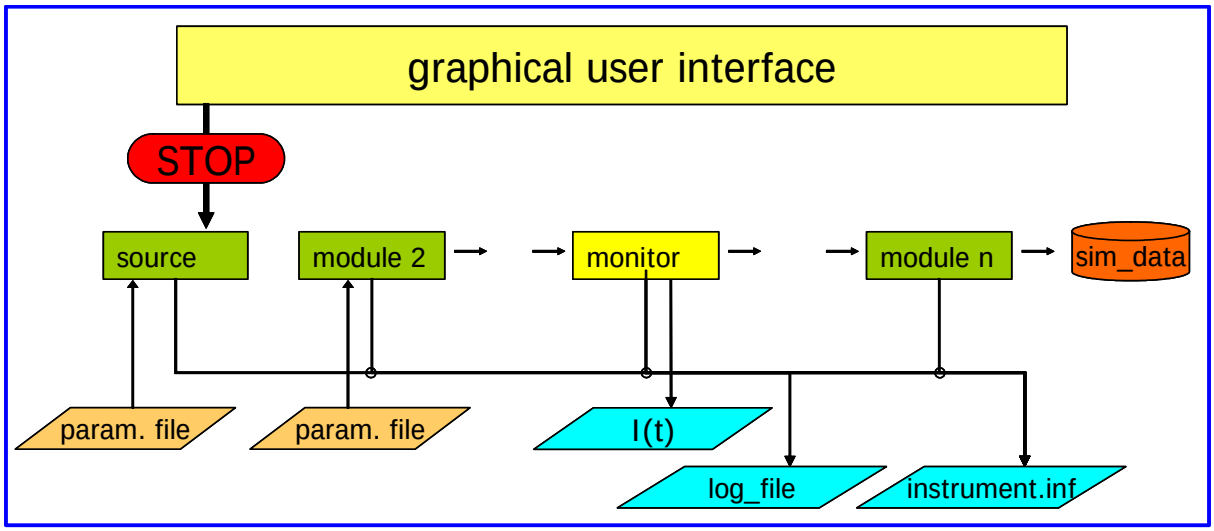


Figure 5: Stopping the simulation without loss of data by 'Soft abort'

6.3 Series of simulations

Since version 2.5 a new option to run a series of simulation is implemented. The series can be defined and started from the graphical user interface. The handling is much more convenient than the previous solution.

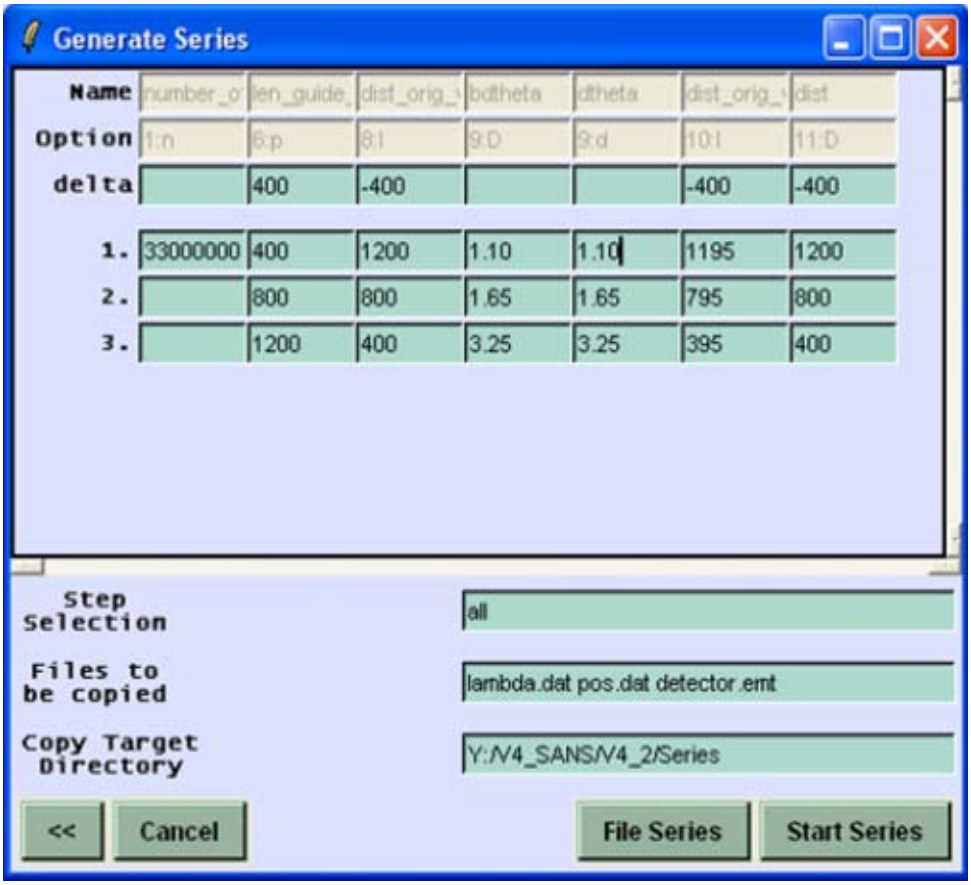


Fig. 6: Window for series of simulations including table of variable parameters. (Here variation of free flight path in a SANS experiment.)

The number of simulation is demanded and the parameters to be changed are defined by just clicking on the item names on the GUI window. Then each variable parameter has to be given in a table for each simulation (s. Fig. 6). If no value is given, the value of the original instrument is used. The same holds for all other parameters.
All files that shall be saved have to be given. After the simulation, they are copied to a target directory with a different prefix (of the file name) for each simulation. The option 'step selection' (see Fig. 6) allows to start only some of the simulations. The whole information about the series is saved with the instrument.

6.4 Ray-tracing options

Altogether there are 3 possibilities to check the proper behaviour of single trajectories (e.g. in case of possible errors). The first is to use the module 'writeout' (at different instrument positions). The parameter sets can be compared and the behaviour of certain trajectories studied.
This procedure is made easier by the ray-tracing option 1 as shown in Fig. 6. In a first run the trajectories that shall be observed are defined, e.g. by using the module 'writeout'. If the run is restarted (in ray-tracing option 1), identical trajectories are created. The only difference is that every trajectory whose ID is found in the 'ray-tracing file' is marked to be observed and a file is opened for each of them. Now each module writes position, divergence, etc. at entrance and exit of the component into this file.
In a second option, only those trajectories whose ID is found in the file are started in the second run. As the ID of every trajectory is checked after the definition of the parameters by MC choices, the initial parameter values of these trajectories are the same as in the first simulation. Thus, the simulation is identical up to a module that uses MC choices. In this module, the sequence of random numbers is redistributed and the simulation runs into a different direction.

The ray-tracing function interprets the first column of the ray-tracing file as list of IDs and does not care about the rest of the file. Therefore, such a file can also be written by hand or any program. Using a file written by 'writeout' is just for convenience.

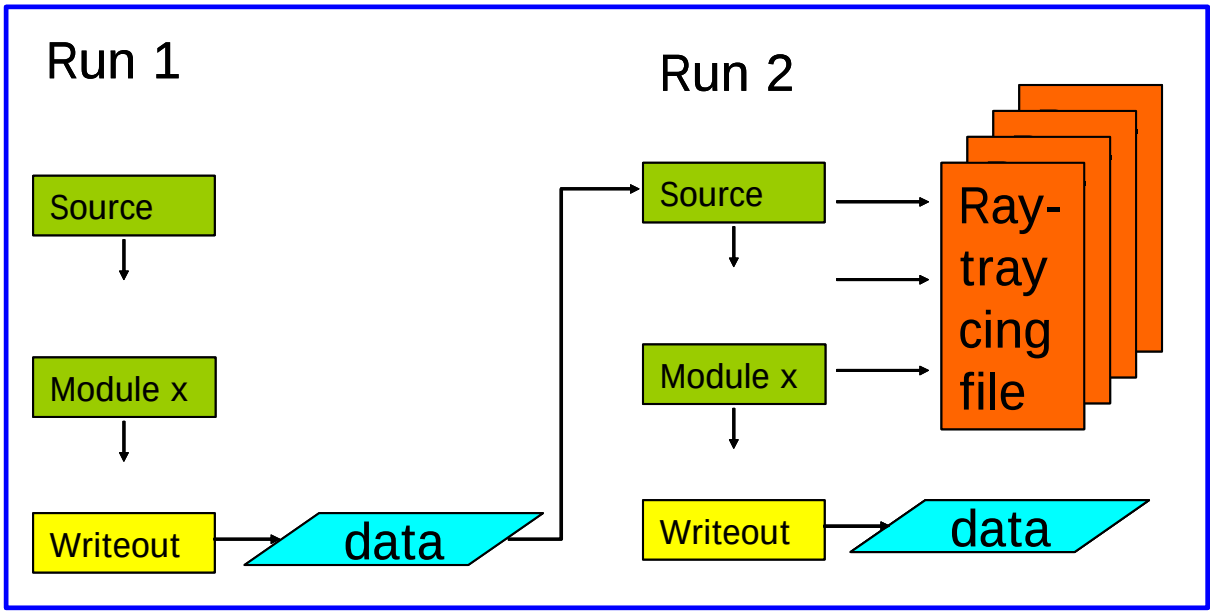


Figure 7: Ray-tracing option 1: writing a data file for each trajectory of interest

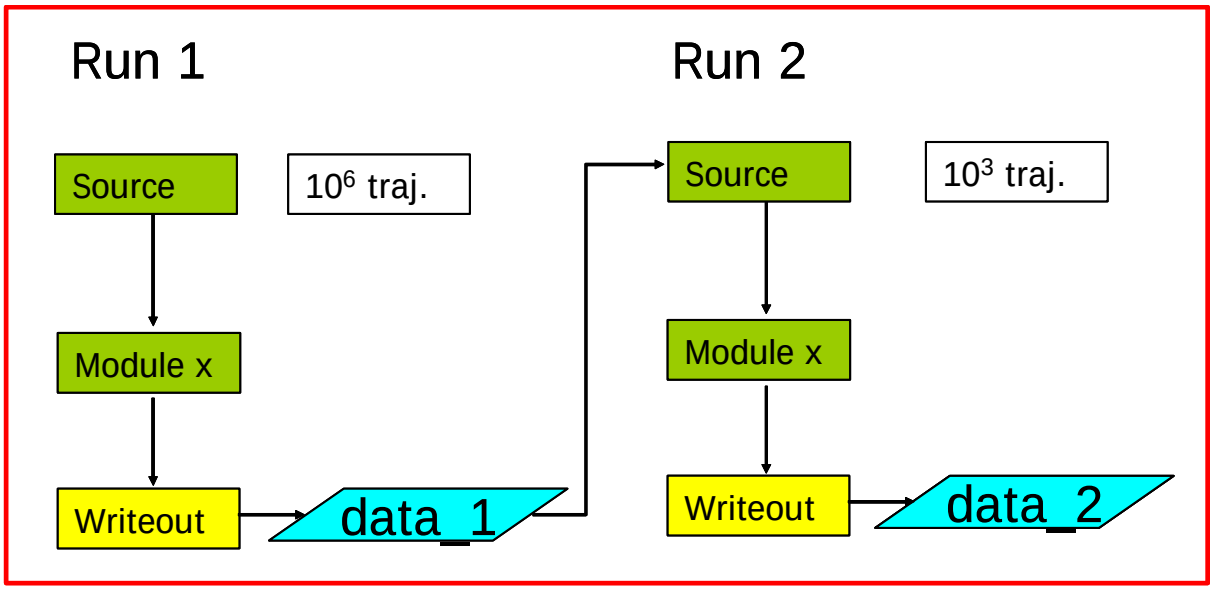


Figure 8: Ray-tracing option 2: starting only trajectories of interest

7. Outlook

The next version will contain an improved detector module, a drawing of the instrument and a tool to find appropriate chopper positions. We also plan to include numerical optimization of instrument parameters into VITESS. Several tests have run well, now we have to find a way to include this in a user-friendly way. Another goal is to realize a more flexible program, e.g. get rid of limitations for number of bins etc. This has already been started and will be continued.
One important task within the MCNSI project is to implement the NeXus data format into VITESS. After some preliminary work, this will be realized this autumn.
Within the MCNSI network, we will take part in code comparison and virtual experiments. While the first code comparison [6] could not be cross-checked by an experiment, this will done with the following comparisons. Now that we are able to simulate an instrument including scattering at a sample and detection, we are not far from the final goal - virtual experiments. Users of neutron facilities should be able to perform a 'virtual experiment' on a computer before they start measuring in order find the best instrument parameters and estimate the measuring time. For a first instrument, this will be done soon in parallel with other packages (supported by MCNSI).
To be able to do that for all existing instruments, we need to complete our set of modules. A sample 'collimation grid' and an 'elliptic mirror' are nearly completed. A 'magnetic hexapole' has already been demanded by users. Additionally, sample modules for radiography and residual stress measurements are missing. Furthermore, a more realistic sample simulation including parasitic Bragg reflection, multiple diffraction and thermal diffuse scattering should be realized in the future.

Acknowledgement

This research project has been supported by the European Commission under the 6th Framework Programme through the Key Action: Strengthening the European Research Area, Research Infrastructures. Contract n°: RII3-CT-2003-505925.

References:

[1] L.L. Daemen, P.A. Seeger, R.P. Hjelm, T.G. Thelliez, Proc SPIE 3771 (1999) 80-89; http://strider.lansce.lanl.gov/NISP/Welcome.html

[2] J. Šaroun and J. Kulda, Physica B 234-236 (1997) 1102-1104; http://omega.ujf.cas.cz/restrax/

[3] K. Nielsen and K. Lefmann, Physica B 283 (2000) 426-432; http://neutron.risoe.dk/mcstas/

[4] G. Zsigmond, K. Lieutenant, F. Mezei, Neutron News 13 No. 4 (2002) 11-14; http://www.hmi.de/projects/ess/vitess/.

[5] W.-T. Lee, X.-L. Wang, J.L. Robertson, F. Klose, Ch. Rehm, Appl. Phys. A 74 [Suppl.] (2002) S1502-S1504; http://www.sns.gov/ideas/.

[6] P.A. Seeger, L.L. Daemon, E. Farhi, W.T. Lee, X.L. Wang, L. Passell, J. Saroun, G. Zsigmond, Neutron News, 13 No. 4 (2002) 24-29.

[7] H. Fritzsche, K. Lieutenant, J. Neutron Research 11, No. 1-2 (2003) 61-68

[8] G. Zsigmond, J. Rodríguez-Carvajal, P.D. Radaelli, K. Lieutenant, F. Mezei, Proc. ICANS-XVI, Eds. G. Mank and H. Conrad (Düsseldorf, Germany, 2003) 541-547.

[9] K. Lieutenant, T. Gutberlet, A. Wiedenmann, F. Mezei, Nucl. Instr. Meth. A, submitted for publication

[10] G. Zsigmond, K. Lieutenant, S. Manoshin, M. Fromme, F. Mezei, Proc. ICANS-XVI, Eds. G. Mank and H. Conrad (Düsseldorf, Germany, 2003) 473-482.

[11] K. Lieutenant, G. Zsigmond, S. Manoshin, M. Fromme, H. N. Bordallo, J. D. M. Champion, J. Peters, F. Mezei, Proc. of the "International Symposium on Optical Science and Technology" (Denver, 2. - 6. Aug. 04), submitted for publication

[12] J. M. Carpenter, E.B. Iverson, D.F.R. Mildner, Nucl. Inst. Meth. Phys. Res, A 483 (2002) 784 - 806.