Draft Set of Standards for Modules in a Monte Carlo Simulation Package

I. Introduction

The general structure of an instrument simulation package was described as illustrated below:

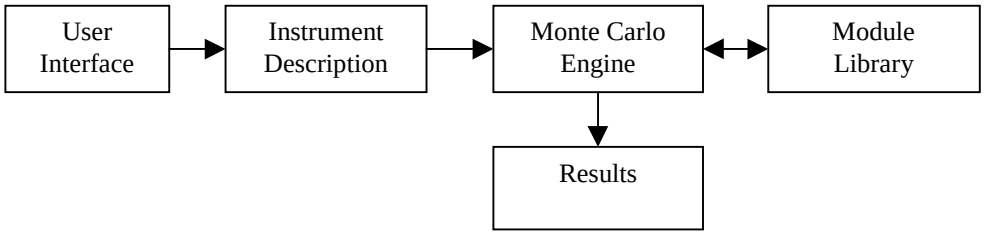

Figure 1. General structure of an instrument simulation package.

The "user interface" provides a description of the instrument to a Monte Carlo simulation "engine". The "engine" propagates neutrons through the model using the "module library" which is a collection of computer codes describing individual components of a neutron scattering instrument. The modules must conform to a common set of standards that allow them to be interchanged among various contributors.

II. Standards

(1) In order for modules to be compatible with a variety of programming environments, the information passed to and from the module must be done in a consistent and transparent fashion. The first part of this standard was an agreement that all variables and parameters that are real numbers provided to and from the modules would be in a double precision format.

(2) The requirement for compatibility with a variety of programming environments also mandates a protocol to be followed for calling the modules. Only a fixed number of objects (see the next section) may be used when calling a module and at least some of these must be of a fixed format so that a variety of "engines" can call the same module. Compatibility among C, C++, F77, and F90 requires that the objects be passed by reference (default in Fortran).

(3) Given this standard for passing parameters and information between the modules and "engine", the next item of discussion concerned the exact information that would be exchanged. The group decided that four objects would allow sufficient communication:

(a) **Neutron** - This object contains a complete description of the neutron as it currently exists within the Monte Carlo simulation in the coordinate system of the module:

Parameter	Data Type	Units
x position	double	m
y position	double	m
z position	double	m
x velocity	double	m/sec
y velocity	double	m/sec
z velocity	double	m/sec
time-of-flight	double	sec
particle mass	double	1 for neutron
charge number	double	0 for neutron
statistical weight	double	fraction
x spin	double	-1 to +1
y spin	double	-1 to +1
z spin	double	-1 to +1
wavelength	double	Å
energy	double	meV

(b) **History** - This object is a compilation of Neutron objects which have been deemed "interesting" according to criteria specified by the author of the module along with some information (module code and event code) to describe the "interesting" event:

Parameter	Data Type	Units
x position	double	m
y position	double	m
z position	double	m
x velocity	double	m/sec
y velocity	double	m/sec
z velocity	double	m/sec
time-of-flight	double	sec
particle mass	double	1 for neutron
charge number	double	0 for neutron
statistical weight	double	fraction
x spin	double	-1 to +1
y spin	double	-1 to +1
z spin	double	-1 to +1
wavelength	double	Å
energy	double	meV
module code	int	index of module that generated this History object
event code	int	key indicating event that triggered the creation of this History object

(c) **Params**- This object contains the list of parameters, which describes the instrument element simulated by the module (e.g. geometry, materials characteristics, or chopper rotation speed). This object will have a different format for each module and its information will be created by the user interface. The module may not modify this array. The author of a module must provide a clear description of these required parameters.

(d) **Fback**- This object contains information, which the module author may choose to provide back to the simulation engine (e.g. the number of times a neutron bounces in a module describing a neutron guide). This object will have a different format for each module. A module does not have to use this facility, but it must provide a placeholder in its list of calling parameters in order to implement a uniform protocol for module calls.