GEANT-4 GPU Port:

Design Document: Detailed Design

Team 8
Stuart Douglas – dougls2
Matthew Pagnan – pagnanmm
Rob Gorrie – gorrierw
Victor Reginato – reginavp

Detailed Design: Version 0
January 11, 2016

Table of Contents

1 Introduction

1.1 Revision History

All major edits to this document will be recorded in the table below.

Table 1: Revision History

Description of Changes	Author	Date
Set up sections and filled out Introduction section	Matthew	2015-12-15
Added sections for Errors and Key Algorithms	Stuart	2016-01-08

1.2 Document Structure & Template

The design documentation for the project is broken into two main documents.

The system architecture document details the system architecture, including an overview of the modules that make up the system, analysis of aspects that are likely and unlikely to change, reasoning behind the high-level decisions, and a table showing how each requirement is addressed in the proposed design.

This detailed design document covers the specifics of several key modules in the project. For each module, an MIS is given fully detailing the interface of the module. Then, the methods for handling errors within the module are discussed, and finally the main algorithms and data structures used by the module are presented.

1.3 List of Tables

1.4 Note About G4 variables

Geant4 uses its own basic types for standard C++ types (G4int, G4bool, G4double, etc). These types are currently just `typedefs` to the respective type as defined in the

system libraries.

2 G4NeutronHPDataPoint

2.1 Description

This class encapsulates all of the data as well as the setter and getter methods that each data point in G4NeutronHPVector's list of data requires. Two private variables are used to store the xSection and the energy of the data point.

2.2 MIS (Module Interface Specification)

2.2.1 Access Program Syntax

Table 2: G4NeutronHPDataPoint – access program syntax

Routine Name	Input	Output	Exceptions
G4NeutronHPDataPoint			
G4NeutronHPDataPoint	G4double, G4double		
=	G4NeutronHPDataPoint		
GetEnergy		G4double	
GetXsection		G4double	
SetEnergy	G4double		
SetXsection	G4double		
GetX		G4double	
GetY		G4double	
SetX	G4double		
SetY	G4double		
SetData	G4double, G4double		

2.2.2 Access Program Semantics

Note that hyphens in routine names and inputs are just for linebreaks due to the table size. The actual routine names and inputs do not have hyphens.

Table 3: NeutronHPDataPoint – access program semantics

Routine Name	Input	Semantics
G4NeutronHPDataPoint		instantiates the class, setting `energy` and `xSec` to 0
G4NeutronHPDataPoint	G4double, G4double	instantiates the class with the inputted `energy` and `xSec`
=	G4NeutronHP-DataPoint	sets the `energy` and `xSec` of the instance to those of the input
GetEnergy		returns `energy` of the instance
GetXsection		returns the `xSec` of the instance
SetEnergy	G4double	sets `energy` of instance to the argument
SetXsection	G4double	sets `xSec` of instance to the argument
GetX		returns the `energy` of the instance
GetY		returns the `xSec` of the instance
SetX	G4double	sets `energy` of instance to the argument
SetY	G4double	sets `xSec` of instance to the argument
SetData	G4double, G4double	sets instance's `energy` and `xSec` to the passed arguments

2.2.3 State Variables

The following variables maintain state for the class, and are all private to the module.

Table 4: G4NeutronHPDataPoint – state variables

Variable	Type	Description
`energy`	G4double	the energy of the particle
`xSec`	G4double	the cross-section of the particle

2.2.4 Environment Variables

There are no environment variables for this module.

2.2.5 Assumptions

It can be assumed that the class will be initialized. As such, all getter methods will return a non-null value.

2.3 Error Handling

This module does not handle errors excplicitly.

2.4 Key Algorithms

This module represents data, and as such does not contain any algorithms.

3 G4NeutronHPVector

3.1 Description

This module stores a large vector of data points (G4NeutronHPDataPoint). It includes functions for setting the data points, retrieving them, and calculating information over them (such as the integral).

3.2 MIS (Module Interface Specification)

Note that hyphens in routine names, inputs, outputs, and exceptions are just for line-breaks due to the table size. The actual routine names, inputs, outputs, and exceptions do not have hyphens.

3.2.1 Access Program Syntax

Table 5: G4NeutronHPVector – access program syntax

Routine Name	Input	Output	Exceptions
G4NeutronHPVector			
G4NeutronHPVector	G4int		
=	G4NeutronHPVector&	G4NeutronHPVector&	
+	G4NeutronHPVector&, G4NeutronHPVector&	G4NeutronHPVector&	
SetVerbose	G4int		
Times	G4double		
SetPoint	G4int, G4NeutronHPDataPoint		
SetData	G4int, G4double,G4double		
SetX	G4int, G4double		
SetEnergy	G4int, G4double		
SetY	G4int, G4double		
SetXsec	G4int, G4double		
GetEnergy	G4int	G4double	
GetXsec	G4int	G4double	
GetXsec	G4double	G4double	
GetXsec	G4double,G4int	G4double	

GetX	G4int	G4double
GetY	G4double	G4double
GetY	G4int	G4double
GetVectorLength		G4int
GetPoint	G4int	const G4NeutronHPDataPoint&
Hash		
ReHash		
InitInterpolation	istream	
Init	istream,G4int, G4double, G4double	
Init	istream, G4double,G4double	
ThinOut	G4double	
SetLabel	G4double	
GetLabel		G4double
CleanUp		
Sample		G4double
Debug		G4double *
Merge	G4NeutronHPVector *, G4NeutronHPVector *	
Merge	G4InterpolationScheme, G4double, G4NeutronHPVector *, G4NeutronHPVector *	
SampleLin		G4double
IntegrateAndNormalise		
Integrate		
GetIntegral		G4double
SetInterpolationManager	const G4InterpolationManager &	
SetInterpolationManager	G4InterpolationManager &	
SetScheme	G4int,const G4InterpolationScheme &	
GetScheme	G4int	G4InterpolationScheme
GetMeanX		G4double
GetBlocked		vector<G4double>
GetBuffered		vector<G4double>

Get15percentBorder		G4double	
Get50percentBorder		G4double	
Check	G4int		G4Hadronic-Exception

3.2.2 Access Program Semantics

Note that hyphens in routine names and inputs are just for linebreaks due to the table size. The actual routine names and inputs do not have hyphens.

Table 6: G4NeutronHPVector – access program semantics

Routine Name	**Input**	**Description**
G4NeutronHPVector		A constructor for the G4NeutronHPVector class. Instantiates the class with no parameters
G4NeutronHPVector	G4int	A constructor for the G4NeutronHPVector class. Instantiates the class with one parameter, determines the value of the nPoints variable based on input
=	G4NeutronHPVector&	Sets the current instance to the passed instance
+	G4NeutronHPVector&, G4NeutronHPVector&	Returns the vector addition of the two passed vectors.
SetVerbose	G4int	sets the Verbose to the input
Times	G4double	Multiplies theData array's y-values by the input, does the same with the corresponding integral
SetPoint	G4int, G4NeutronHP-DataPoint	sets theData[G4int] to the passed in DataPoint
SetData	G4int, G4double,G4double	sets theData[G4int] to the passed in G4double values
SetX	G4int, G4double	sets theData[G4int]'x value to the passed in G4double value
SetEnergy	G4int, G4double	sets theData[G4int]'s x value to the passed in G4double value
SetY	G4int, G4double	sets theData[G4int]'s y value to the passed in G4double value
SetXsec	G4int, G4double	sets theData[G4int]'s y value to the passed in G4double value
GetEnergy	G4int	Returns theData[G4int]'s current x value

GetXsec	G4int	Returns theData[G4int]'s current x value
GetXsec	G4double	Returns theData's y value corresponding to the x section starting at G4int
GetX	G4int	Returns theData[G4int]'s current x value
GetY	G4int	Returns theData[G4int]'s current y value
GetVectorLength		Returns the length of the vector, which is stored in nEntries
GetPoint	G4int	Returns theData[G4int]'s current value
Hash		sets theHash's value based on theData
ReHash		clears theHash and re-evaluates it based on theData
InitInterpolation	istream	calls Init on the istream input
Init	istream,G4int, G4double, G4double	Initializes the data in theHash of length G4int
Init	istream, G4double,G4double	Initializes the data in theHash of length that is read in from a file
ThinOut	G4double	Thins out theData() by removing unnecessary elements and calls ReHash
SetLabel	G4double	Sets the Label value to G4Double
GetLabel		Returns the current Label value
CleanUp		Sets the length of theData to 0, clears the integral and clears theHash
Sample		Samples X according to distribution Y
Debug		Returns theIntegral
Merge	G4NeutronHPVector*, G4NeutronHPVector*	interpolate between labels, continue in unknown areas by subtraction of the last difference.
Merge	G4Interpolation-Scheme, G4double, G4NeutronHPVector*, G4NeutronHPVector*	interpolate between labels according to G4InterpolationScheme, cut at G4double, continue in unknown areas by subtraction of the last difference.
SampleLin		Samples X according to distribution Y, linear int
IntegrateAndNormalise		Calculates the integral for every entry in theIntegral and normalizes each calculation
Integrate		Calculates the integral for every entry in theIntegral
GetIntegral		Linear interpolation over theIntegral
+SetInterpolation-Manager	const G4Interpolation-Manager&	sets theManager to the input

SetInterpolation-Manager	G4Interpolation-Manager&	sets theManager to the input
SetScheme	G4int, G4Interpolation-Scheme&	Appends the G4Interpolation-Scheme to theManager
GetScheme	G4int	Returns the current G4Interpolation-Scheme associated with theManager
GetMeanX		Returns the average x value of all data points in theData
GetBlocked		Returns the current value of theBlocked
GetBuffered		Returns the current value of theBuffered
Get15percentBorder		gets the integral from each data point to the last data point and returns the first one within 15% of the last data point
Get50percentBorder		gets the integral from each data point to the last data point and returns the first one within 50% of the last data point
Check	G4int	checks that passed index is greater than the number of points, throwing an exception if not

3.2.3 State Variables

The following variables maintain state for the class, and are all private to the class. Note that hyphens in variable names and types are just for linebreaks due to the table size. The actual variable names and types do not have hyphens.

Table 7: G4NeutronHPVector – state variables

Variable	Type	Description
theLin	G4NeutronHP-Interpolator	the linear interpolator for sampling data
totalIntegral	G4double	integral over all data points from theData
theData	G4NeutronHP-DataPoint*	array of G4NeutronHPDataPoint, stores all data points in vector
theManager	G4Interpolation-Manager	manages the interpolation schemes, knows how to interpolate data
theIntegral	G4double*	array of integrals where theIntegral[i] is the integral of all data points from theData up until i
nEntries	G4int	the number of data points to consider when performing calculations over theData
nPoints	G4int	the number of data points in theData
label	G4double	number
theInt	G4Neutron-Interpolator	the interpolator for sampling data (may not be linear)
Verbose	G4int	verbosity level, some statements will only print to console with higher values
isFreed	G4int	only used for debugging, 1 if class has been destructed 0 otherwise
theHash	G4NeutronHP-Hash	stores the x and y value of every tenth data point from theData to speed up getting the minimum index in theData of a data point with X larger than a given value
maxValue	G4double	maximum value of Xsec or Y passed in SetData, SetY, or SetXSec so far. Initialized to -DBL_MAX (min representable double).
theBlocked	vector <G4double>	deprecated: vector still exists in class but data never added to it
theBuffered	vector <G4double>	stores buffer of samples to speed up sampling the vector
the15percent-BorderCash	G4double	the X value of the first data point with an integral no more than 15% smaller than the integral of the last data point
the50percent-BorderCash	G4double	the X value of the first data point with an integral no more than 50% smaller than the integral of the last data point

3.2.4 Environment Variables

There are no environment variables for this module.

3.2.5 Assumptions

It can be assumed that the module will be initialized before other functions are called.

3.3 Error Handling

The `Check` method throws a G4HadronicException on error, however it is the only function to do so in the module. In the other functions, erroneous input is not handled explicitly beyond control statement checks that will assume default values for any invalid parameters.

3.4 Key Algorithms

There are a variety of algorithms used in the module. When porting to the GPU, the same algorithms will be modified to run in parallel. In general, this consists of taking array traversals and running the procedures executed sequentially at the same time on different cores of the GPU.

4 CMake Files

4.1 Description

The current build system used by Geant4 is CMake, consisting of *CMakeLists* text files in each source code directory detailing the files to compile and link, and further compiler directives. The user calls the `cmake` program with arguments (such as *useCuda*) for the build to generate the necessary makefiles. Support for CUDA and the *nvcc* CUDA compiler are built in to CMake. Although not a module in the traditional sense, CMake will still be the basis for enabling and disabling GPU functionality, and was included for that reason.

4.2 MIS (Module Interface Specification)

4.2.1 Access Program Syntax

CUDA support is built in to CMake, as such no new access programs or public macros will be created.

4.2.2 Access Program Semantics

CUDA support is built in to CMake, as such no new access programs or public macros will be created.

4.2.3 State Variables

Table 8: CMake Files – state variables

Variable	Type	Description
`useCuda`	Boolean	if set to true, the makefiles generated by CMake will include directives to compile and link the CUDA code and will execute ported procedures on the GPU. Default is false.

4.2.4 Environment Variables

- CUDA source files (.cu) containing the GPU code. CMake files will contain directives to compile and link the CUDA files.
- Source code from Geant4 project, such as the G4NeutronHPVector.cpp file. The relevant source code files will be compiled and linked as per CMake directives to the CUDA files listed above.

4.2.5 Assumptions

It is assumed the user has CMake installed, as it is required for Geant4.

4.3 Error Handling

If the tries to enable CUDA without compatible hardware, CMake will detect this and output a fatal error message. The user will not be able to enable CUDA unless they have compatible hardware. If the user is using an older version of CMake (before 2.8) that does not support CUDA compilation, a fatal error message will be outputted.

4.4 Key Algorithms

CMake is the existing build system for generating make files for the project. As such, there are no key algorithms to document.