G4-STORK: A Geant4 Based Reactor Kinetics Simulation Code

User Manual rev. 0.5

L. Russell and A. Buijs

McMaster University, Department of Engineering Physics
Hamilton, Canada, L8S 4L7

and

G. Jonkmans

Atomic Energy of Canada Limited, Chalk River Labs
Chalk River, Canada, K0J 1P0

Contents

1. Installation

G4-STORK is distributed as source code for Linux operating systems and the latest C++ compilers. Users are required to compile the source code on their system using the provided make utilities (cmake and make files). G4-STORK should also work on Windows, although this option has never been tested and some features may not work due to incompatible third-party software (parallel processing).

1.1 Installing required software

G4-STORK is based on Geant4, and thus, requires access to the Geant4 source code and data libraries. However, several additional programs are needed to use all the features of G4-STORK. In particular, the parallel processing requires TOP-C and Marshalgen, two parallel processing utilities developed by Gene Cooperman.

1.1.1 Geant 4.9.6

G4-STORK requires the latest version of Geant4, version 4.9.6. The source code and data libraries can be downloaded from the Geant4 website (url below). The website also features several user guides for installation, application development and more advanced topics. G4-STORK users should take advantage of these resources to familiarize themselves with Geant4 before using G4-STORK.

Website: `http://geant4.cern.ch/`
User Support: `http://geant4.cern.ch/support/index.shtml`
Installation Guide: `http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/InstallationGuide/html/index.html`

1.1.1.1 Geant4 data libraries

The only Geant4 data library that is necessary for G4-STORK is G4NDL 4.2 (Geant4 Neutron Data Library). However, Geant4 can automatically download and install all of the major data libraries during installation. This is the simplest method to obtain the data libraries and gives users access to all of the Geant4 physics models.

G4-STORK can also use custom built libraries that follow the G4NDL format. This allows users to develop libraries with neutron cross sections at temperatures greater than 0 K (the evaluation temperature of the G4NDL libraries). This can significantly reduce the on-flight Doppler broadening used in Geant4 for the resonances in the cross section data. These libraries need to be named so that their name ends with the evaluation temperature of the cross sections. A data file evaluated at 293.6 K would end in "_T293.6".

e.g. UserNeutronHPData_T293.6

1.1.2 TOP-C 2.5.2

TOP-C is a parallel processing utility created by Gene Cooperman of Northeastern University. TOP-C consists of a series of C++ header files and libraries that can used to easily enable Geant4 based programs (and other programs) to run in parallel. TOP-C uses MPI (Message Passing Interface) pipes and a master-slave architecture to spread computations across multiple processing units.

Installation of TOP-C is relatively simple. The libraries and header files simply need to be placed in locations where G4-STORK can locate them. By default, TOP-C will install the headers and libraries in */usr/local/include* and */usr/local/bin* respectively. TOP-C may be downloaded from the TOP-C website listed below.

Website: `http://www.ccs.neu.edu/home/gene/topc.html`

1.1.3 Marshalgen 1.0

Marshalgen is another parallel processing utility created by Gene Cooperman. Since MPI pipes transmit serial data, any C++ data (objects) transferred between different processors must be convert into, and reconstituted from, serial data. This process is known as marshalling and unmarshalling respectively. Marshalgen simplifies this process by automatically producing marshallable versions of existing C++ classes. The user must simply indicate with comments how the data members of the class are accessed (see StorkNeutronData.hh).

Marshalgen is not strictly necessary to run G4-STORK since the marshalled classes have already been created. However, if the user wants to change any of these classes, Marshalgen will be required to remake the marshalled classes.

Marshalgen can be found at the website below. Additionally, Marshalgen requires the parsing utilities *Bison* and *Flex* which can easily be installed through a Linux package manager (e.g. yum, apt-get, Synaptic Package Manager).

Website: `http://www.ccs.neu.edu/home/gene/marshalgen.html`

NOTE: The Marshalgen source code contains a syntax error that inhibits its use for certain data types. The file *marshalgen-1.0/Phase1/mgen.pl* contains the line

Listing 1.1: Marshalgen syntax error (mgen.pl)

```
683    $annot.= "\t\t\tcopy_off \+\= sizeof\($strElementType\)\);\n";
```

The error in Listing 1.1 is that line 683 contains an extra ' \)'. The correct code is shown in Listing 1.2. For simplicity, a fixed copy of Marshalgen has been included with the G4-STORK code.

Listing 1.2: Fixed code (mgen.pl)

```
683    $annot.= "\t\t\tcopy_off \+\= sizeof\($strElementType\);\n";
```

1.2 Installing G4-STORK

Use the following steps to compile G4-STORK

1. Unpack the G4-STORK source code tarball into the folder of your choice (the folder should be accessible without root access)

   [bash]$ tar -xzvf ${G4-STORK_TARBALL} ${DESTINATION}

2. Move to the build folder

   [bash]$ cd Build

3. Create and configure the build of G4-STORK using the cmake file (CMakeLists.txt).

   [bash]$ cmake -DGeant4_DIR=${GEANT4_INSTALL}/lib64/Geant4-9.6.1/Geant4Config.cmake ../

   Note: ${GEANT4_INSTALL} is the path to the installed location of Geant4, and on 32-bit systems users would use "lib" instead of "lib64" (see Section 3 of the Geant4 installation manual). Additionally, "../" denotes the path to the G4-STORK source code, and could be different if you use a different build folder.

4. Compile G4-STORK using make.

   [bash]$ make -j*N*

   Note: *N* represents the number of processes make will use to compile the code.

5. (Optional) Install G4-STORK to bin folder of the Geant4 install directory.

   [bash]$ make install

This creates a sequential (non-parallel) executable of G4-STORK in the build folder named *g4stork*. The following build options can be used to customize your build of G4-STORK

- **-DTOPC_USE=1**
  Creates a parallel version of G4-STORK using TOP-C
- **-DTOPC_USE_SEQ=1**
  Creates a sequential version of G4-STORK using TOP-C (not parallel, requires -DTOPC_USE=1)
- **-DG4VISUALIZE_USE=1**
  Uses a Geant4 viewer to show geometry and particle tracks. Only use with small simulations (few primaries and runs).
- **-DCMAKE_BUILD_TYPE=Debug**
  Compile G4-STORK with debugging flags.

- **-DG4VERBOSE_TRACKING_USE**
  Diagnostic tool that shows each step every particle takes.
- **-DG4TIME_USE**
  Diagnostic tool that breaks down the computation time for each step of the simulation.
- **-DSTORK_EXPLICIT_LOSS**
  Diagnostic tool that outputs the type of interaction for neutrons lost in an event (i.e. fission, capture, escape).

2. Running G4-STORK

2.1 Recompiling G4-STORK

G4-STORK needs to be recompiled if the G4-STORK code is modified, or if additional code is added, such as new simulation geometries. The user simply needs to repeat the steps in Section 1.2.

1. If additional files are being added, the makefile must be recreated using cmake

   [bash]$ cmake -DGeant4_DIR=${GEANT4_INSTALL}/lib64/Geant4-9.6.1/Geant4Config.cmake ../

2. Before recompilation, the user can clean up the old executable and object files, but this is normally not necessary.

   [bash]$ make clean

3. Then, G4-STORK is recompiled using the make command

   [bash]$ make -j*N*

2.2 Setup

Before G4-STORK can run, the user must set the environment variable pointing to the location of the neutron cross section data (i.e. the high precision neutron data libraries in the G4NDL format). This can be done manually by setting the environment variable itself, or by using the Geant4 configuration script.

Manual

Set the environment variable G4NEUTRONHPDATA to point to the location of a G4NDL formatted, high precision neutron library. This method should be used if you are not using the standard G4NDL libraries.

[bash]$ export G4NEUTRONHPDATA=${NEUTRON_HP_DATAPATH}

Geant4 Config Script

Run the Geant4 script file *geant4.(c)sh*. This file is automatically generated during the installation of Geant4 and it contains the location of the standard Geant4 data libraries (assuming that they were installed during the installation).

[bash]$ source ${GEANT4_INSTALL}/bin/geant4.sh

2.3 Running a simulation

To run a simple, sequential simulation, simply run the G4-STORK executable in the build directory with an input file (e.g. Input.txt).

```
[bash]$ ./g4stork Input.txt
```

For a sequential simulation, all of the simulations options are controlled by the input file (see Appendix A for a full list of inputs and Appendix B for example input files).

2.3.1 Running a parallel simulation

To run a parallel simulation, the following steps need to be taken first

1. Rebuild the makefile using the "-DTOPC_USE=1" option with cmake, and recompile G4-STORK.
2. Set up a processor listing file (default name is *procgroup*, see Appendix C). If the default name is used, simply put this file in the same directory as the G4-STORK executable. If another name is used, or the processor listing file is not in the same folder as the executable, use the

   --TOPC-procgroup=${processor_listing_path}

   option when running G4-STORK (see below). In general, the number of slave processes in the processor listing file is the number of slaves that will be used by G4-STORK.
3. Set the input file so that the number of events per run is greater than the number of slaves. G4-STORK runs most efficiently if the number of events per run is a multiple of the number of slaves since each slave handles a single event at a time.

A parallel simulation may now be started using

```
[bash]$ ./g4stork ${input_file}
```

if the processor listing file is named *procgroup* and is in the same folder as the executable. Otherwise, use

```
[bash]$ ./g4stork --TOPC-procgroup=${processor_listing_path} ${input_file}
```

A. G4-STORK Input Parameters

A.1 Required parameters

The following parameters must be defined for every simulation.

WORLD {string}

Set the world type as *Sphere*, *C6Lattice*, *Cube* or a user-defined geometry.

NUM_RUNS {integer}

Set the number of runs in the simulation.

NUM_EVENTS {integer}

Set the number of events per run.

NUM_PRIMARY_PER_EVENT {integer}

Set the number of primary neutrons that are initialized at the beginning of each event.

RUN_DURATION {double (ns)}

Set the duration of the run in nanoseconds.

A.2 Geometric and material parameters

These parameters control the geometry and materials of the simulation. Specific properties of materials in the simulation geometry are identified using material-property (mat-prop) pairs. The possible materials, properties and default units of the properties are

mat = {all, fuel, coolant, moderator, poison}
prop = {temperature, density, dimension, concentration}
units = {K, g/cm^3, cm, percent}

When using these in the input file, at least the first 3 letters of each material and property name must be used.

MAT {integer}

Set the homogeneous material of the *Sphere* or *Cube* worlds.

SET_WORLD_PROP {mat} {prop} {double}

Set the value of a material-property (in the default units of the property).

INTERPOLATION_DATA {mat} {prop} {string}

Set the data file (file name) that contains the time-dependent data profile for the material-property. This file is used to create a time-dependent simulation geometry (see Appendix D). The INTERPOLATION_DATA parameter can be repeated multiple times to make other material-property pairs time-dependent.

A.3 Shannon entropy parameters

The Shannon entropy is used to track the convergence of the neutron and fission site distributions.

SE_MESH {integer} {integer} {integer}

Set the resolution of the mesh used to calculate the Shannon entropy in the x, y and z directions. The dimensions of the mesh are the smallest box that contains the simulation geometry, and is subdivided by the meshing parameters given by this parameter. The default values are (20, 20, 20).

SE_NUM_RUNS_CONV {integer}

Set the period over which the Shannon entropy must show convergence before the source and neutron distributions are considered converged. The default value is 25 runs.

SE_CONV_LIMIT {double}

Set the convergence limit of the Shannon entropy in percent. The Shannon entropy from every run during the convergence period must be within the convergence limit to every other entropy value for convergence to be achieved. The default value is 2%.

A.4 Simulation options

These parameters control different options for the G4-STORK simulation. The bool values that need to be used in the input file are **0** (*false* or *off*) and **1** (*true* or *on*).

SEED {long integer}

Set the seed for the Geant4 random number generator. Either setting the seed to 0 or not using this parameter in the input file will cause G4-STORK to use a random seed.

PERIODIC_BC {bool}

Creates periodic boundaries at the edge of the simulation geometry. This option is used for the infinite lattice geometries (*C6Lattice* and *Cube*) and is 1 (*on*) by default for these geometries. Otherwise, the default value is 0 (*off*).

RENORMALIZE {bool}

Turn on/off the renormalization of the neutron survivors at the end of each run. The default value is 1 (*on*).

INSTANT_DELAYED {bool}

Produce the delayed neutrons instantly as prompt neutrons; the delayed neutrons still use the delayed neutron distributions for energy and momentum. The default value is 0 (*off*).

UNIFORM_DISTRIBUTION {bool}

Spread the initially generated primary neutrons uniformly across the world. The default value is 0 (*off*).

INTERP_START_COND {bool}

Start the time-dependent material and geometry changes immediately at the beginning of the simulation. Normally, the simulation waits until the source and neutron distributions have converged before starting the interpolation. The default value is 0 (*wait until convergence*).

A.5 File input and output parameters

These parameters set and control the input and output files used by G4-STORK. If these parameters are not set, no files are used for either input or output.

OUTPUT_LOG {string}

The name (and path) of the output file used by G4-STORK to log the simulation results.

OUTPUT_SOURCE {string}

The name (and path) of the output file used by G4-STORK to record the neutron survivors at the end of specific runs.

OUTPUT_SRC_FREQ {integer}

The frequency, in number of runs, at which the neutron survivors are recorded to the source file (if the parameter OUTPUT_SOURCE is set). The default value is the total number of runs in the simulation (e.g. record the distribution only once). Note that the last distribution is always recorded if OUTPUT_SOURCE is set regardless of the frequency.

INITIAL_SRC_FILE {integer}

The name (and path) of the input file used to set the initial primaries and initial delayed neutrons. If this option is used, NEUTRON_ENERGY and UNIFORM_DISTRIBUTION are ignored. This option can be used to continue a previous simulation.

B. Example Input Files

Listing B.1: U235 sphere input file

```
# Example input for a simplified Godiva
# Pure U235 sphere
# k_eff ~= 1.0


# Build a homogeneous sphere
WORLD Sphere

# Set sphere radius to 8.7 cm
SET_WORLD_PROP all dimension 8.7

# Number of runs, events per run and primaries per event
NUM_RUNS 100
NUM_EVENTS 4
NUM_PRIMARY_PER_EVENT 1250

# Initial random seed
SEED 2013092304

# Produce delayed neutrons instantaneously
INSTANT_DELAYED 1

# Duration of each run (ns)
RUN_DURATION 25

# Initial neutron energy (MeV)
NEUTRON_ENERGY 2.0

# Output files
#OUTPUT_LOG Log_U235_Sphere.txt
#OUTPUT_SOURCE Src_U235_Sphere.txt

# Frequency of source output
#OUTPUT_SRC_FREQ 50
```

Listing B.2: Uranium-heavy-water sphere input file

```
# Example input for a UHW sphere
# Mixture of natural uranium (10% w/w) and heavy water (D20)
# k_eff ~= 1.0


# Build a homogeneous sphere
WORLD Sphere

# Use UHW material
MAT 2

# Set sphere radius to 87.5 cm
SET_WORLD_PROP all dimension 87.5

# Number of runs, events per run and primaries per event
NUM_RUNS 100
NUM_EVENTS 4
NUM_PRIMARY_PER_EVENT 1250

# Initial random seed
SEED 2013092304

# Produce delayed neutrons instantaneously
INSTANT_DELAYED 1

# Duration of each run (ns)
RUN_DURATION 5e5

# Initial neutron energy (MeV)
NEUTRON_ENERGY 2.0

# Output files
OUTPUT_LOG Log_UHW_Sphere.txt
OUTPUT_SOURCE Src_UHW_Sphere.txt

# Frequency of source output
OUTPUT_SRC_FREQ 50
```

Listing B.3: CANDU 6 lattice cell input file

```
# Example input for an infinite lattice of CANDU 6 lattice cells
# Pressure tube reactor
# Fresh fuel (natural uranium)
# Heavy water coolant and moderator

# Build a C6 lattice cell (with periodic boundary conditions by default)
WORLD C6Lattice

# Number of runs, events per run and primaries per event
NUM_RUNS 100
NUM_EVENTS 4
NUM_PRIMARY_PER_EVENT 1250

# Initial random seed
SEED 2013092304

# Produce delayed neutrons instantaneously
INSTANT_DELAYED 1

# Duration of each run (ns)
RUN_DURATION 5e5

# Initial neutron energy (MeV)
NEUTRON_ENERGY 2.0

# Output files
OUTPUT_LOG Log_CANDU6.txt
OUTPUT_SOURCE Src_CANDU6.txt

# Frequency of source output
OUTPUT_SRC_FREQ 50
```

C. Example Processor Listing File

Listing C.1: Processor listing file

```
1 # Procgroup file
2 # Creates two slaves that run the G4-STORK executable in the same folder as
3 # the procgroup file.
4
5 local 0
6 localhost 1 ./g4stork
7 localhost 1 ./g4stork
```

The processor listing file is used to specify the number of slaves and the characteristics of each slave. The file has the following

- Comment lines starting with '#'.
- Master declaration
  local 0
- Slave declarations
  localhost 1 ./g4stork

The slave declaration has the form

{slave address} 1 {executable command} >{output file}

where the arguments are as follows

1st Argument - Address of the slave. For processors on the same computer use "localhost".

2nd Argument - The command used by the slave process to execute G4-STORK. This argument can be replaced by "-", which causes the slave to execute the same command as the master.

3rd Argument - (Optional) Standard output of the slave is redirected to this file.

D. Example Interpolation Data File

Listing D.1: UHW Temperature Data

```
1 1
2 3 2
3 3
4 0.0 293.6
5 1.0e7 293.6
6 6.0e7 1000.0
```

The interpolation data files are used to control time-dependent material-properties. The format of these files is based on the format necessary for the *G4InterpolationManager* class in Geant4. The file is split into two sections: the configuration parameters and the data. The configuration section has four parameters in general:

1. Number of ranges - This is usually one. If it is not one, then the configuration and data sections become more complex.
2. Number of data points in the range
3. Interpolation scheme - Linear-linear is 2. Enumeration of the interpolation schemes can be found in the *G4InterpolationManager* class.
4. Number of data points in the range (again)

The data section is simply a two dimensional table of data where 'x' values (first column) are time values and 'y' values (second column) are the values of the given property at that time. The data table should have the dimensions {number of data points in range} $\times$ 2.