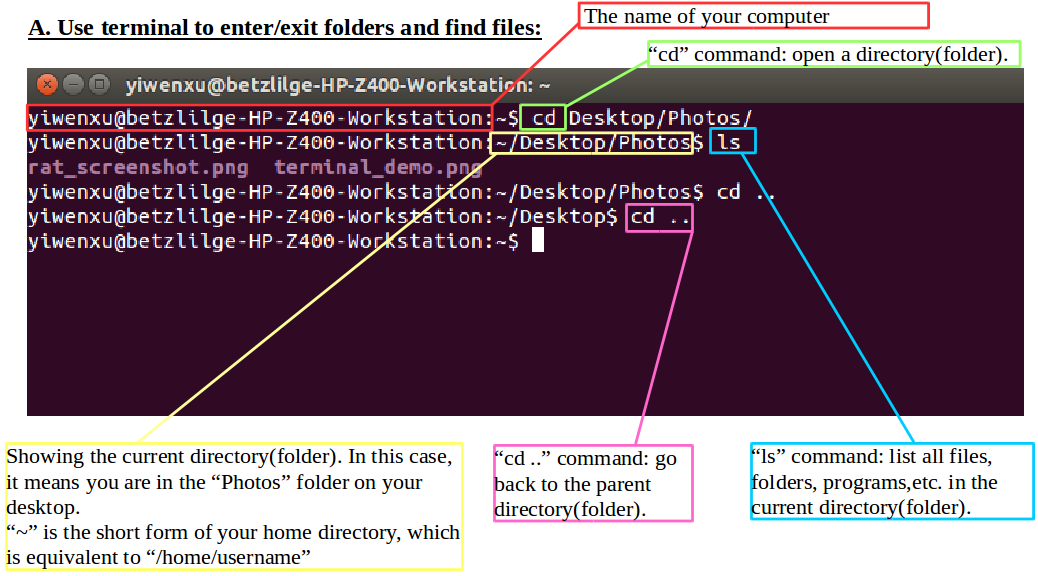
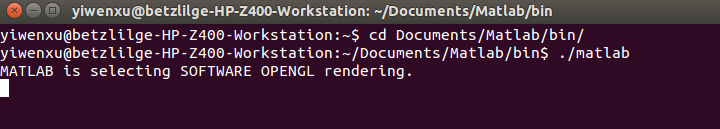
**Appendix I – Linux Basics**

****

**B. Use terminal to execute programs:**

Example: open a software named “matlab”



step1: use “cd” command to enter the folder containing matlab.

step2: use “./”+“software name” to execute the program. In this case type “./matlab” in the terminal and hit

**Appendix II – Building From Source**

**A. Why building from source?**

Binary packages: - packages that are already built from source by someone else

- easy to install

- may not have all options from upstream package

Source packages: - take more time to install

- heavy customization option

**B. How to build software/library from source?**

(The following is a an outline for building using Cmake. Please see below the installation of MeshTool as a detailed example.)

step 1: Download source code or clone the source code from Github.

step 2: Go to the directory where you downloaded the source code file in terminal

step 3: Use command “tar zxvf” + “<file name.tar.gz>” to extract the file into a folder. The command may be different for different types of archive files.eg:



step 4: Enter the folder in terminal and create a build-release folder inside it (for the purpose of out-of-source build).

step 5: Enter the build-release folder and run Cmake using command “ccmake ..”

step 6: Configure using Cmake and generate the software/library.

step 7: Make install the software/library.

**C. What is Cmake?**

CMake is an open-source, cross-platform family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in the compiler environment of your choice. The suite of CMake tools were created by Kitware in response to the need for a powerful, cross-platform build environment for open-source projects such as ITK and VTK.

We use Cmake to build software/library from source.

**FullMonte Software Installation Instruction**

**Step 1: Install Itk-snap and Paraview**

**Step 2: Install MeshTool**

**Step 3: Install FullMonteSW**

**Step 1: Install Itk-snap and Paraview**



**Itk-snap:**

Itk-snap is a software application used to segment structures in 3D medical images.

Download link:<http://www.itksnap.org/pmwiki/pmwiki.php?n=Downloads.SNAP3>

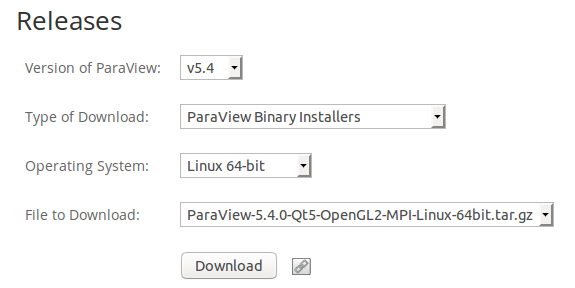
Version: 3.6.0 or later

Tutorial:<http://www.itksnap.org/pmwiki/pmwiki.php?n=Train.Sep2014>

**Paraview:**

ParaView is an open-source, multi-platform data analysis and visualization application. ParaView users can quickly build visualizations to analyze their data using qualitative and quantitative techniques. The data exploration can be done interactively in 3D or programmatically using ParaView’s batch processing capabilities.

Download link:**<https://www.paraview.org/download/>**

[](https://www.paraview.org/download/)

Version:

(5.4 or later)

Tutorial:<https://www.paraview.org/tutorials/>

**Step 2: Install MeshTool**

**I. Download/build All Pre-Requisites**

Download Cmake from:<https://cmake.org/download/>. Version 3.1 or later

Download Qt from :<https://download.qt.io/archive/qt/5.3/5.3.2/>.

Download VTK from:<http://www.vtk.org/download/>. Version 6.3.0 or later.

Download Boost from:<http://www.boost.org/>. Recent Version.

Download CGAL from:<https://github.com/CGAL/cgal/releases/tag/releases/CGAL-4.8.1>. Choose “Source Code (tar.gz)”, and build from source. See a detailed building instruction here:<http://doc.cgal.org/latest/Manual/installation.html>.

Download Eigen Library from:<http://eigen.tuxfamily.org/index.php?title=Main_Page>. Choose latest version and build from source using Cmake.

**II. Clone source code of MeshTool and FullMonteSW from Github:**

~$ git clone http://github.com/zachzzc/MeshTool

~$ cd MeshTool/

~/MeshTool$ mkdir build-release

~/MeshTool$ cd ..

~$ git clone http://github.com/jeffreycassidy/FullMonteSW

~$ cd MeshTool

~/MeshTool$ cd build-release/

~/MeshTool/build-release$ ln -s ~/FullMonteSW/ FullMonte

~/MeshTool/build-release$ ccmake ..

**Interpretation of the commands above:**

line 1: Clone the MeshTool software package from Github (need internet connection)

line 2: Enter the directory of the package that is cloned onto your computer

line 3: Create a new directory “build-release” inside MeshTool. This directory is where the software will be finally built.

line 4: Go back to home directory

line 5: Clone the FullMonte software package from Github (need internet connection)

line 6: Enter MeshTool directory

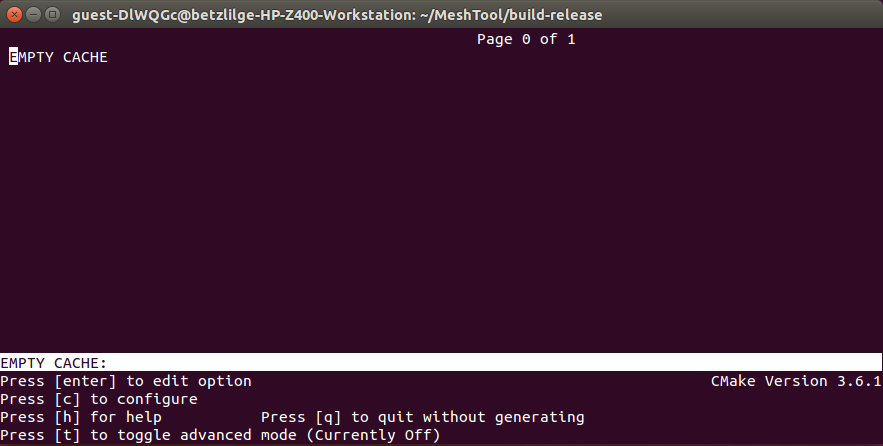
line 7: Enter build-release directory (you can combine these two steps into one by using “cd MeshTool/build-release/”

line 8: Make connection between MeshTool and FullMonteSW. To allow MeshTool run properly, FullMonteSW is needed.

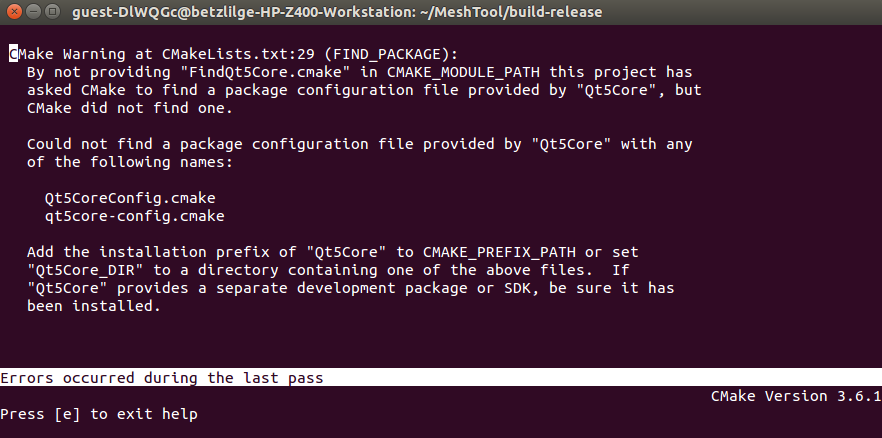
line 9: Open CMake Gui

**III. Configure and build MeshTool:**

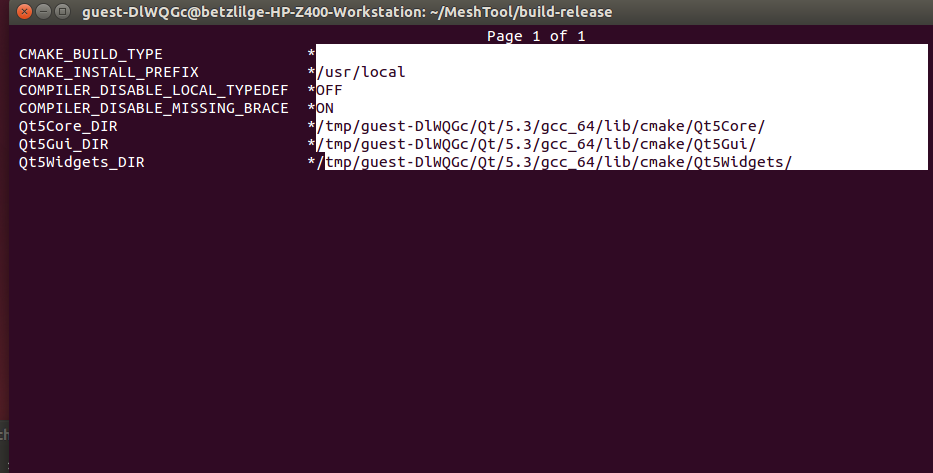
After typing the last command “ccmake ..” in your terminal, you should be able to see a new window that looks like this:



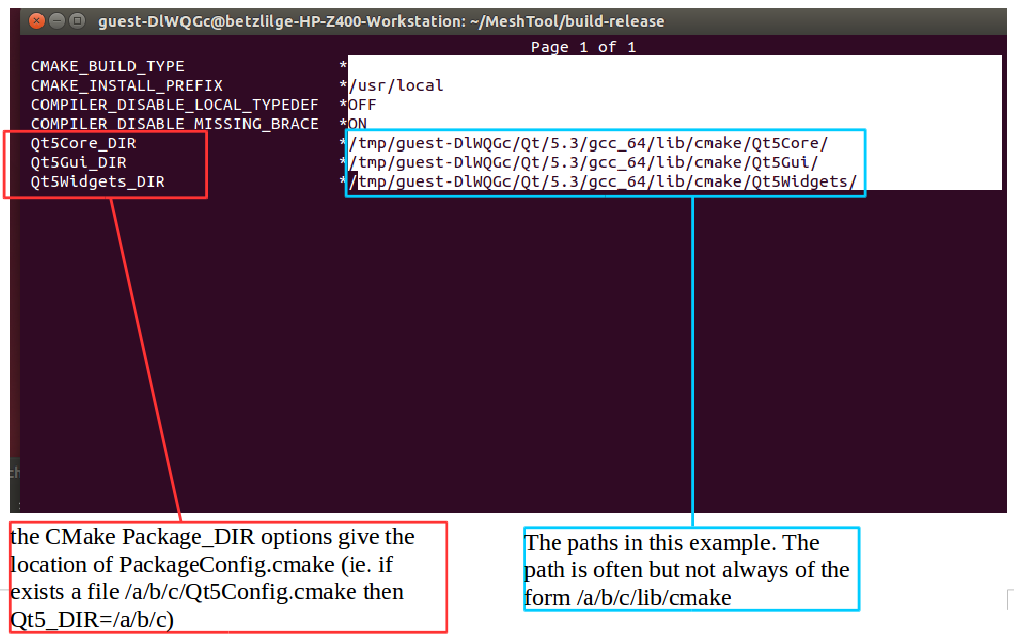
Press [c] to configure and you will see some error messages. Most of the time, these errors need to be fixed before generating the software.



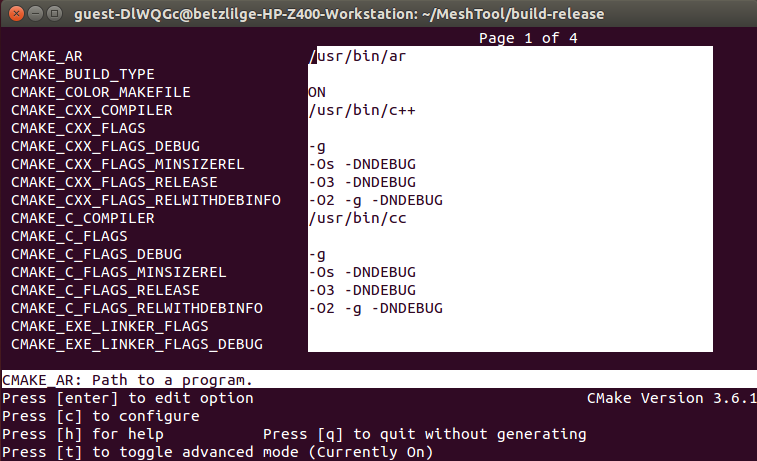
Press [e] to go back to the previous window and you will see some entries:

****

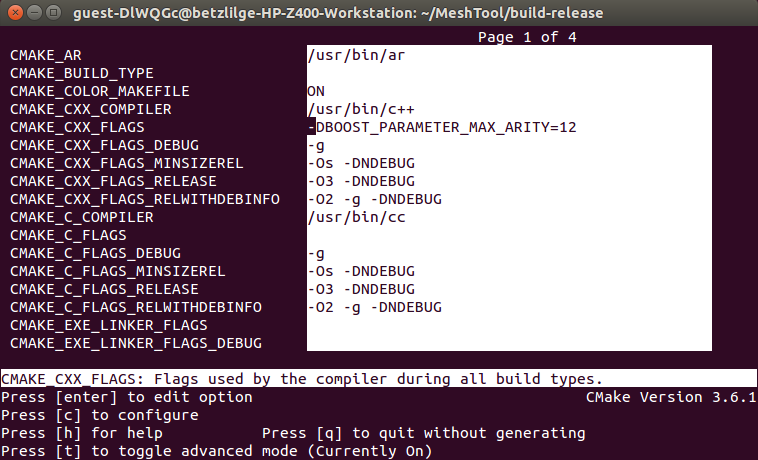
To fix the error messages in this example, we need to setup all Qt paths that Cmake failed to find manually. Use the arrow keys to choose the directory that you want to change (in this case they are “Qt5Core\_DIR, Qt5Gui\_DIR, Qt5Widgets\_DIR”). Press [enter] to allow edition. And set the path according to the following (Press [enter] again once finish editing):

****

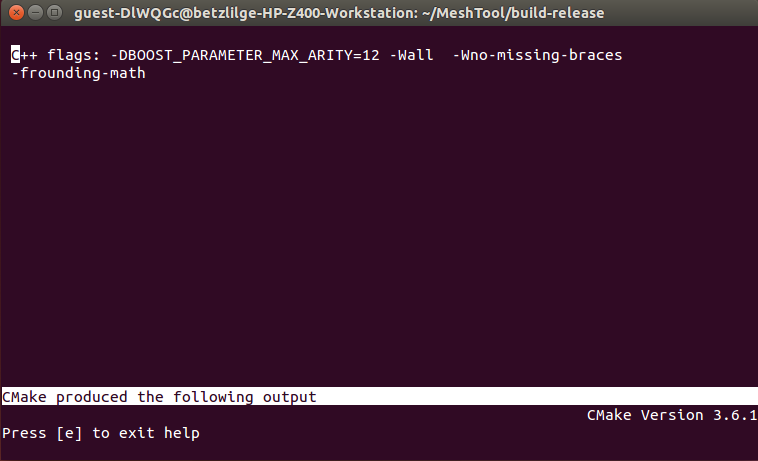
Then press [t] to toggle advanced mode. You will see more options as shown below:

****

Once in the advanced mode, set “CMAKE\_CXX\_FLAGS” to “-DBOOST\_PARAMETER\_MAX\_ARITY=12”. And then press [c] to configure again.

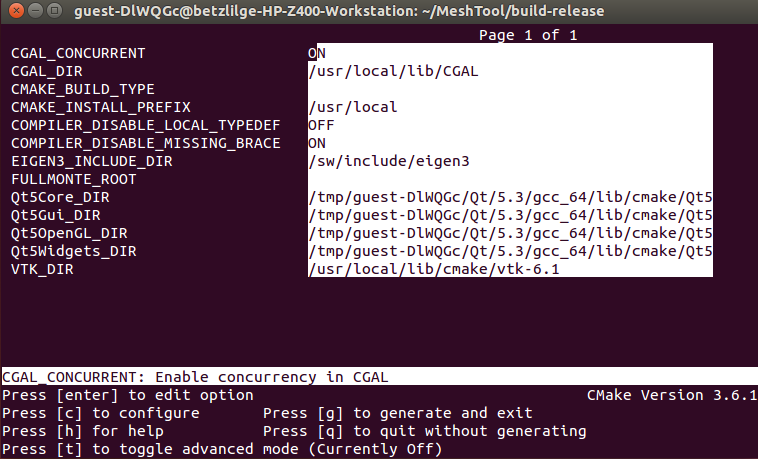
****

If you then see this message or nothing, press [e] and you are almost there. If there is any other error message, please fix it before you proceed to the next step.

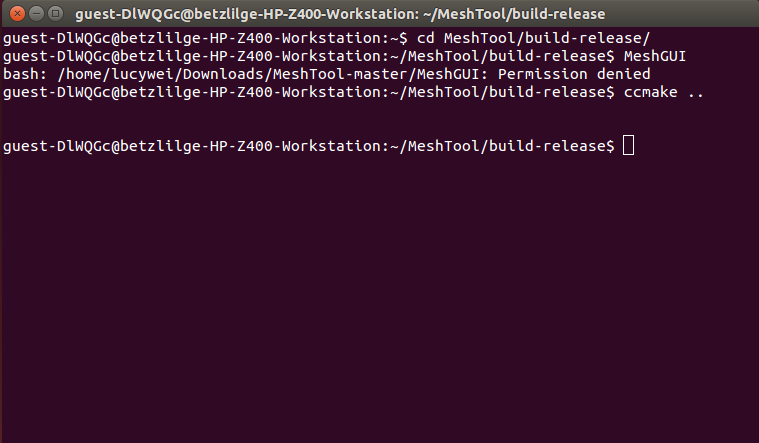
****

You should be able to see a new option [g] by now(If you don't, try to configure one more time). Press

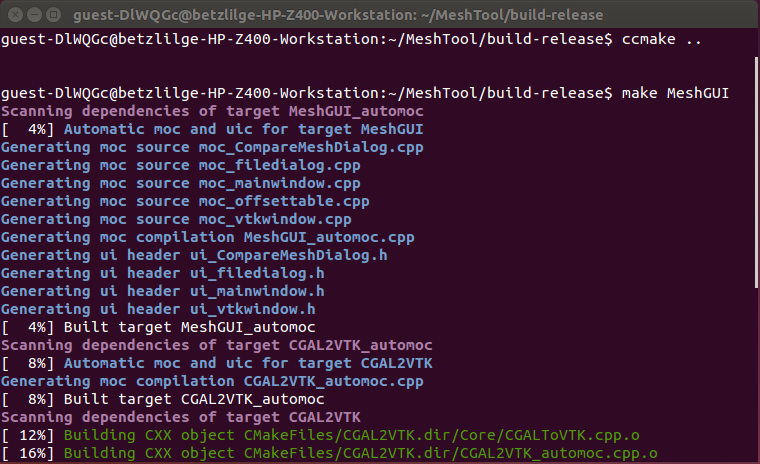
[g] to generate the software.

****

Now the window should have disappeared and you are back to the terminal.

****

Type “make MeshGUI” in the terminal and the software will then be generated automatically. Once you see [100%], the software is built successfully.

****

**Step 3: Install FullMonteSW**

Option 1: Run FullMonteSW using virtual machine

A virtual machine running Ubuntu 16.04 is set up with FullMonteSW installed and built. To access this virtual machine, Oracle VM VirtualBox is required. It can be downloaded at: <https://www.virtualbox.org/wiki/Downloads>

Locate the FullMonteSW.vmdk image file for the virtual machine. Open and clone the virtual machine use VirtualBox.

Note that the image file is quite large (~10GB) and the virtual machine should be given as much disk space as possible. Under the current setting (2 cores, 4096 MB base memory, 32 GB dynamic storage), some of the graphics feature cannot be fully displayed.

Option 2: Install from source

1. System Pre-Requisites (in addition to those required for MeshTool)

* Modern compiler with C++11 support: G++ version 4.9 or higher, or Clang.

If you have a recent Linux distribution, a newer version of G++ should have already been installed. To check the G++ version on your machine, in terminal, run: g++ --version.

Instruction for installing G++ 4.9: <https://askubuntu.com/questions/428198/getting-installing-gcc-g-4-9-on-ubuntu>.

* Tcl 8.5 or 8.6 (for Tcl wrapping)

May need development headers (tcl8.5-dev and tk-dev packages)

Download Tk package: sudo apt-get install tk-dev

Download Tcl package: sudo apt-get install tcl8.6-dev

* SWIG 3.0 (for Tcl wrapping)

Download SWIG: sudo apt-get install swig3.0

1. Configure and Build FullMonte

Make sure you read the notes below and the file README.md in the FullMonteSW package first before configuring or building the software.

The configuring and building process of FullMonte is similiar to that of MeshTool. An out-of-source build is recommended, meaning that the object files are created in a folder separate from the source files. A typical developer’s use case would have separate Debug and Release builds.

Additional notes:

* It may be necessary to specify paths to packages such as Boost (-DBOOST\_ROOT). CMake documentation will help with standard packages (VTK, Tcl/Tk, Swig) and compiler settings.
* To enable TCL and VTK interfaces, the WRAP\_VTK and WRAP\_TCL values must be set to ON (default is OFF). Users must also modify the paths for external softwares/libraries to local paths on their machine, especially when the install location is non-standard.
* Ensure that build type is set to release for full performance. Architecture should be set appropriately to "AVX" or "AVX2" (Use 'cpuid' command to help choosing AVX vs AVX2)

**The processor must support at least AVX.**

Table 1. Important variables

|  |  |  |
| --- | --- | --- |
| Variable | Value | Function |
| BOOST\_ROOT | path | Location of Boost library and header files |
| WRAP\_VTK | ON/OFF | Generate VTK interface code and scripting support |
| WRAP\_TCL | ON/OFF | Generate Tcl interface code |
| CMAKE\_BUILD\_TYPE | Release/Debug | Build type (use Release for best performance) |
| VTK\_DIR | path | Folder containing VTKConfig.cmake |

An example build script, ‘build\_fullmonte.sh’, is provided with the FullMonte software package. By changing the variable paths to local path, the above mentioned steps will be completed automatically.

For FullMonte configuration and build on Mac OS, consult README.md available in the FullMonteSW package.

For further information on cmake, consult documentation on<https://cmake.org>.

**Step 4: Run Simulation**

There are several example scripts provided with the FullMonte software package. They can be found under ‘/FullMonte\_build\_dir/Examples’.

To run these examples,

1. Go to ‘/FullMonte\_build\_dir/bin’ directory.
2. Type ‘./tclmonte.sh ../Examples/name\_of\_example’ at command line.
3. Resultant files are placed either in /bin or the respective /Example directory.

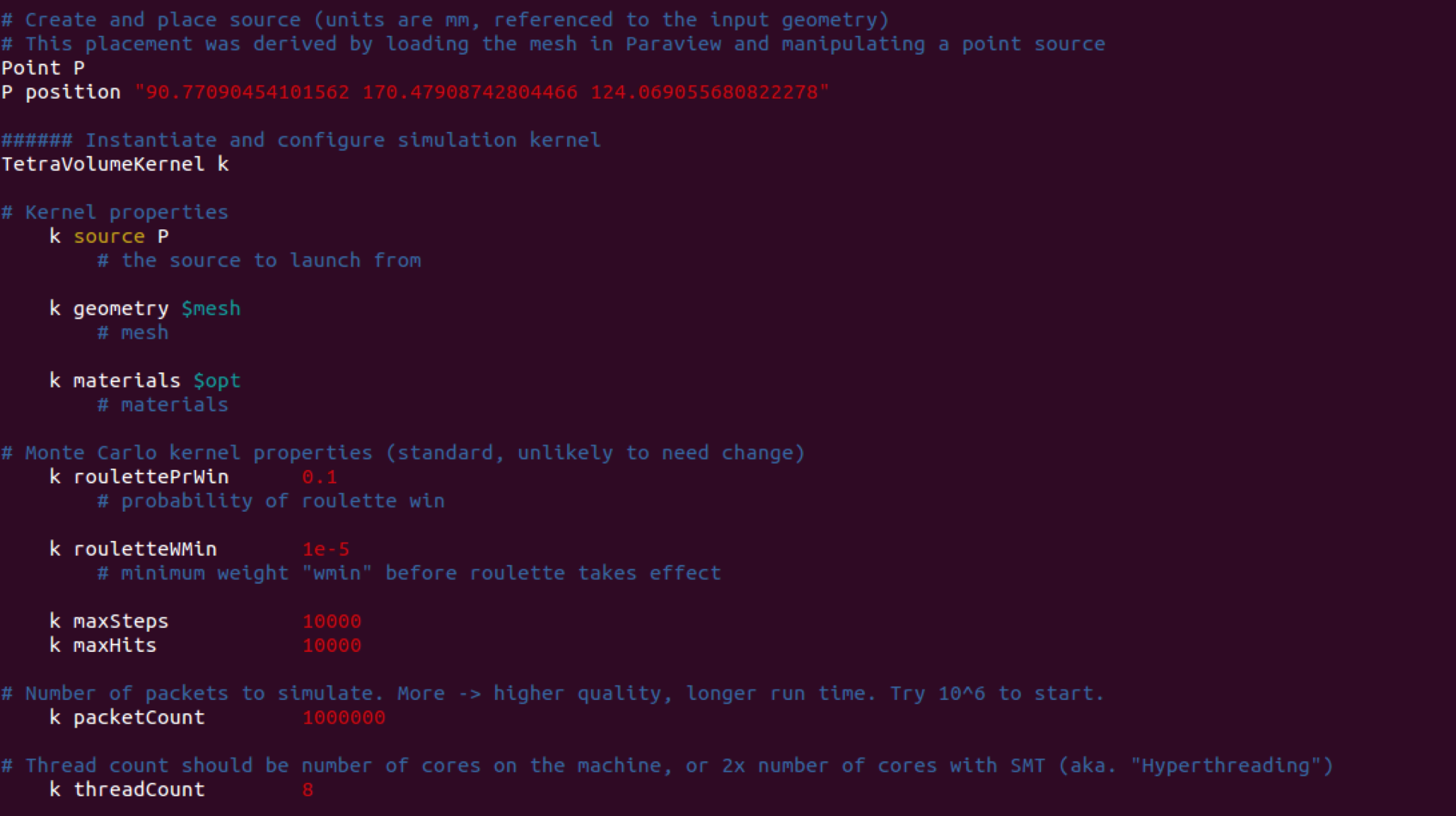
**Tutorial on Running Colin27 Examples and Regression Tests**

Colin27 is a human brain mesh. Its corresponding mesh and tissue property files are placed under /FullMonte\_source\_dir/data/MMC/Colin27. In the FullMonte\_build\_dir, there are several examples available:

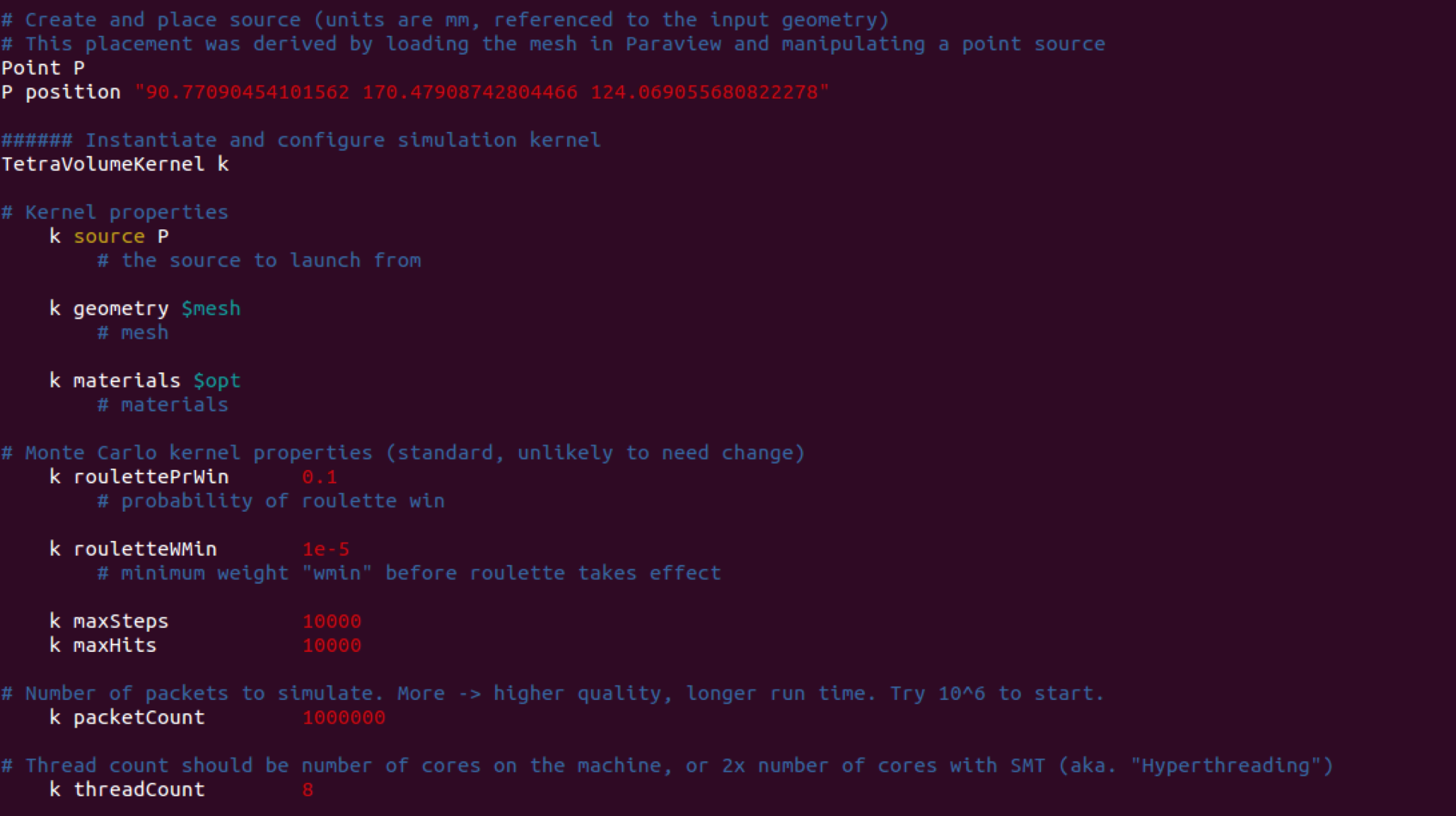
1. Run FullMonte simulation given a source and output result in terms of fluence/energy.
2. Compare FullMonte simulation results with MMC\*.
3. Perform regression tests to verify correctness of the latest FullMonte result with saved golden copies.

1. **Run FullMonte simulation with a point source and output energy data.**

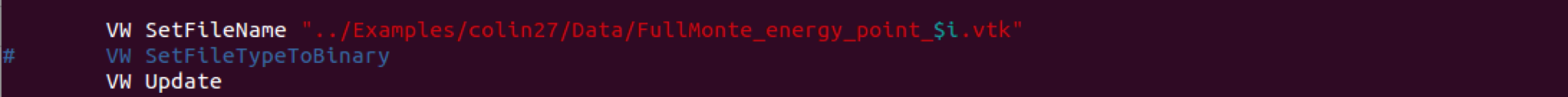
**Step 1:** Open /FullMonte\_build\_dir/Examples/colin27\_energy.tcl and modify the file.



Set the type and location of the source. In this case, a point source is specified at location: (x, y, z) = (90.77090454101562, 170.47908742804466, 124.069055680822278). The unit is millimeter (mm).



Specify the standard kernel properties and most are not expected to change. The example shown here uses a kernel that outputs data summed over the volume of each tetrahedron. Some key parameters that could be modified include ‘source’, which is the source type used, and ‘packetCount’, which is the number of photon packets to launch.



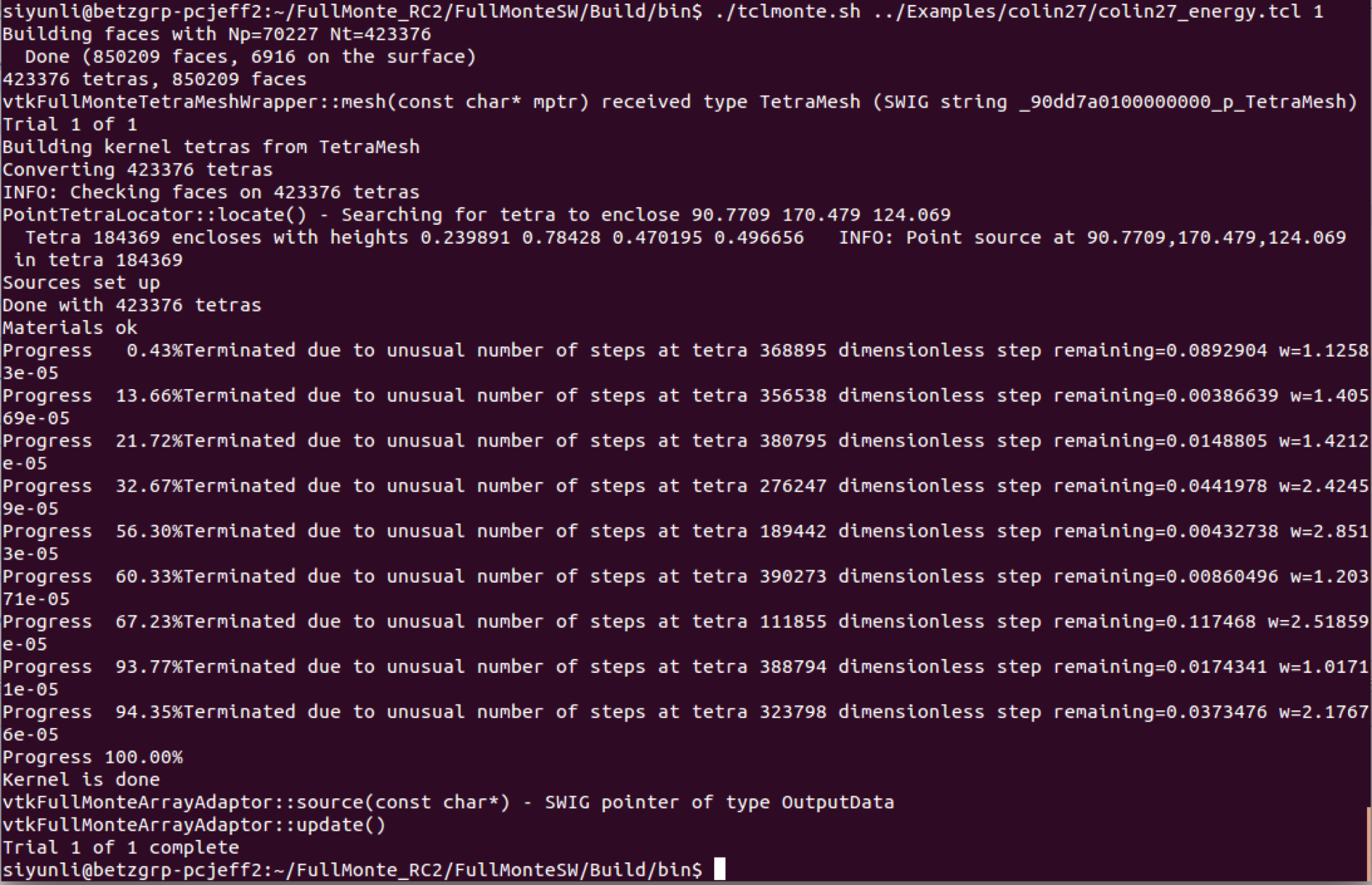
Set the name of the output file. The naming convention is: **Simulator\_outputType\_sourceType\_trialNumber.vtk.**

An example would be: “FullMonte\_energy\_point\_0.vtk”.

**Step 2:** Change directory to /FullMonte\_build\_dir/bin and run:

./tclmonte.sh ../Examples/colin27\_energy.tcl 1

This will run the simulator only once. To run multiple trials, change the last command line option to any desired number.



The simulation should run and display messages similar to the figure above. Note that there would be some messages like “Terminated due to unusual number of steps…”. These messages should not be a concern as they simply mean that the roulette is initiated and the packet is terminated. The output file is saved as “FullMonte\_energy\_point\_0.vtk” under the directory: /FullMonte\_build\_dir/Examples/Data/colin27.

1. **View simulation results.**

**Step 0:** Complete Example I to generate the ‘FullMonte\_energy\_point\_0.vtk’ file for visualization.

If ParaView is installed (Highly Recommended):

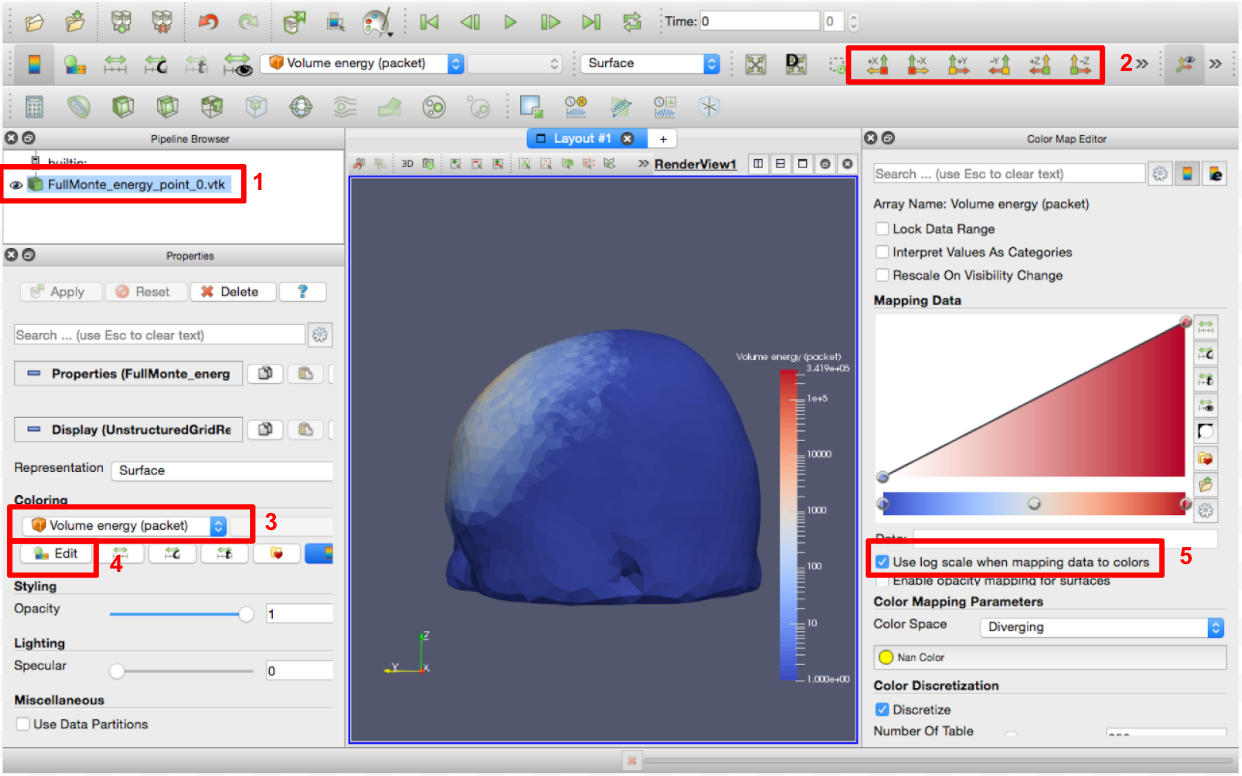
**Step 1:** In ParaView, go to ‘File’ -> ‘Open’ to load the file ‘FullMonte\_energy\_point\_0.vtk’. The opened file should be in ‘Pipeline Browser’ and make sure that the eye symbol on the left is ON.

**Step 2:** Click the first button of the panel shown to set view direction to positive x-axis. One may experiment with other view directions using the other buttons.

**Step 3:** Change the ‘Coloring’ to ‘Volume energy (packet)’ to display the simulation result.

**Step 4:** By default, ParaView uses linear scale to map data to colours. In this example, the range of the energy data is quite large so a logarithmic scale is more appropriate to use. Click on the ‘Edit’ button to open up the ‘Color Map Editor’.

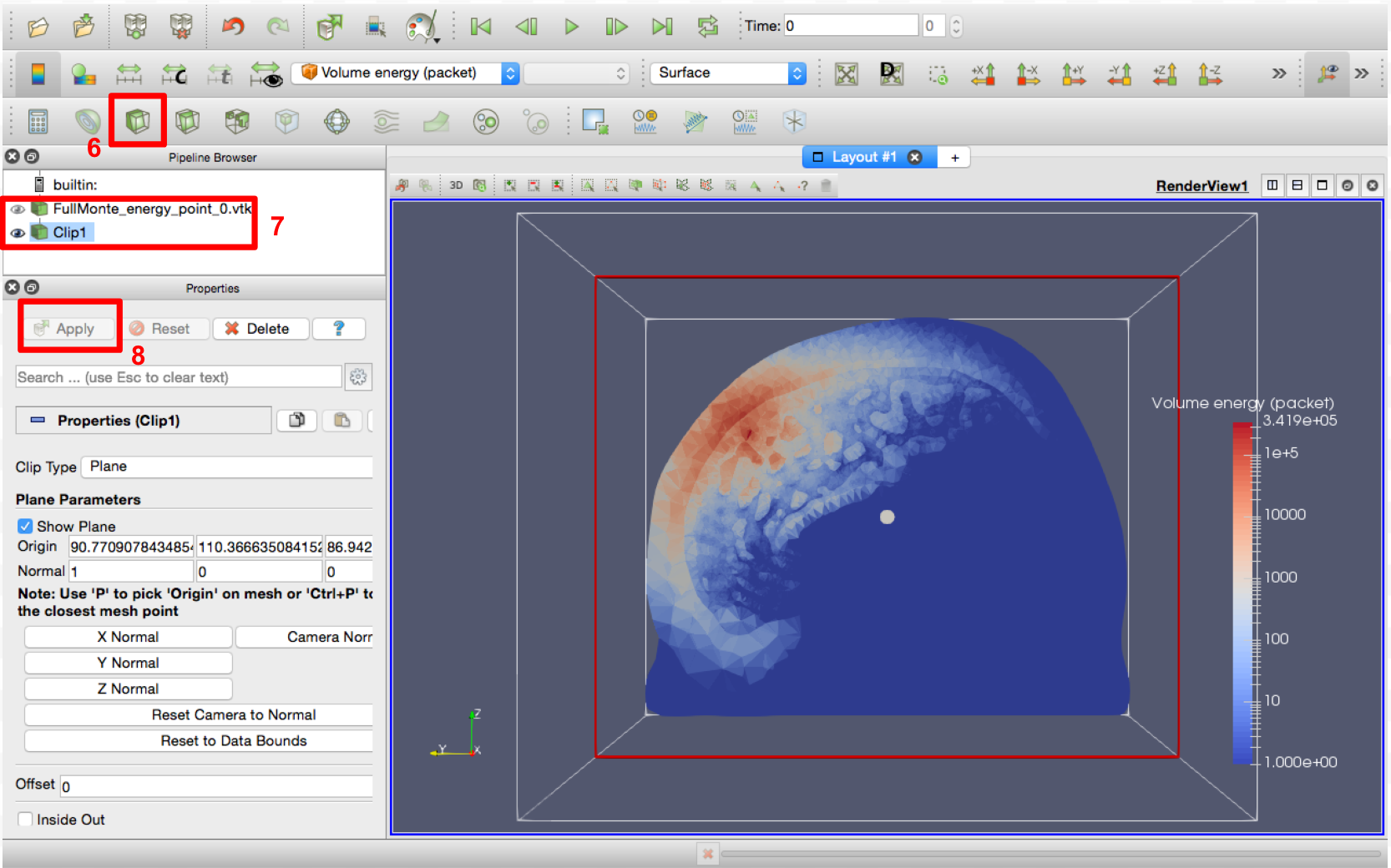
**Step 5:** In the ‘Color Map Editor’, check the ‘Use log scale when mapping data to colors’ option.



**Step 6:** To view the inside of the mesh model, one can click on the ‘Clip’ tool.

**Step 7:** A clip should be created in the ‘Pipeline Browser’. Make sure to turn off the eye symbol beside ‘FullMonte\_energy\_point\_0.vtk’ and turn on the one beside ‘Clip 1’.

**Step 8:** In the ‘Properties’ window, the ‘Apply’ button should be green. Click on ‘Apply’ to apply the changes to the rendered view.



Some basic interactive tools include:

* To rotate the rendered view, click and drag with the left key of the mouse.
* To zoom in/out the rendered view, scroll up/down using the scroll wheel on the mouse.
* To move the mesh model, press and drag using the scroll wheel on the mouse.

ParaView offers a wide variety of filters that can be used to view the data. For further information, please consult the documentation for the proper version of ParaView at: <http://www.vtk.org/download/>

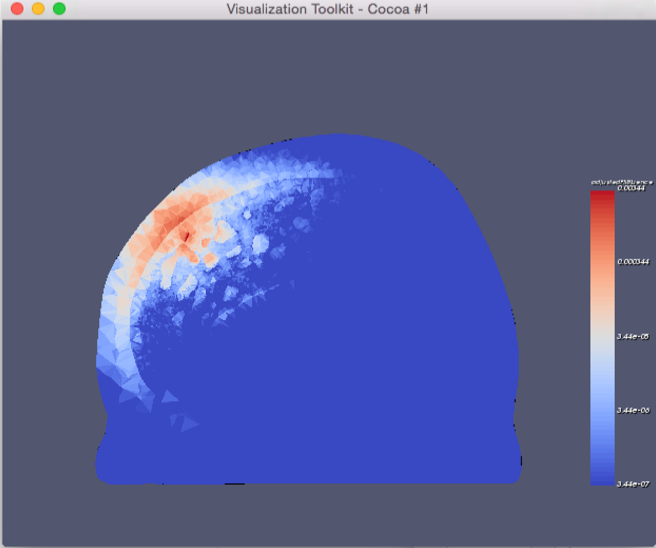
If using the provided C++ class in the FullMonteSW package, to generate an OpenGL window:

**Step 1:** Change directory to /FullMonte\_build\_dir/Examples/colin27/FullMonte\_visualize/. Make sure that FullMonte\_visualize.in has dataType = energy and sourceType = point.

**Step 2:** In terminal, run the pre-compiled executable:

./FullMonte\_visualize FullMonte\_visualize.in

**Step 3:** The interactive window as shown below should automatically pop up. Use the mouse to rotate/zoom/pan the model.



**Step 4:** A visualization figure is saved in .png format. File name: FullMonte\_energy\_point.png

To open the plot file from terminal, type: xdg-open fileName. This has the same effect as double-clicking in a file window.

1. **Compare FullMonte with MMC, both using point sources and outputting energy.**

**Step 0:** Complete Example I to generate the ‘FullMonte\_energy\_point\_0.vtk’ file for visualization.

**Step 1:** Change directory to /FullMonte\_build\_dir/Examples/colin27/FullMonte\_MMC\_point/. Make sure that FullMonte\_MMC\_point.in contains the following information:

1 //N

1e6 //packets

1e5 //topNvalues

1e-8 //threshold

energy //dataType

point //souceType

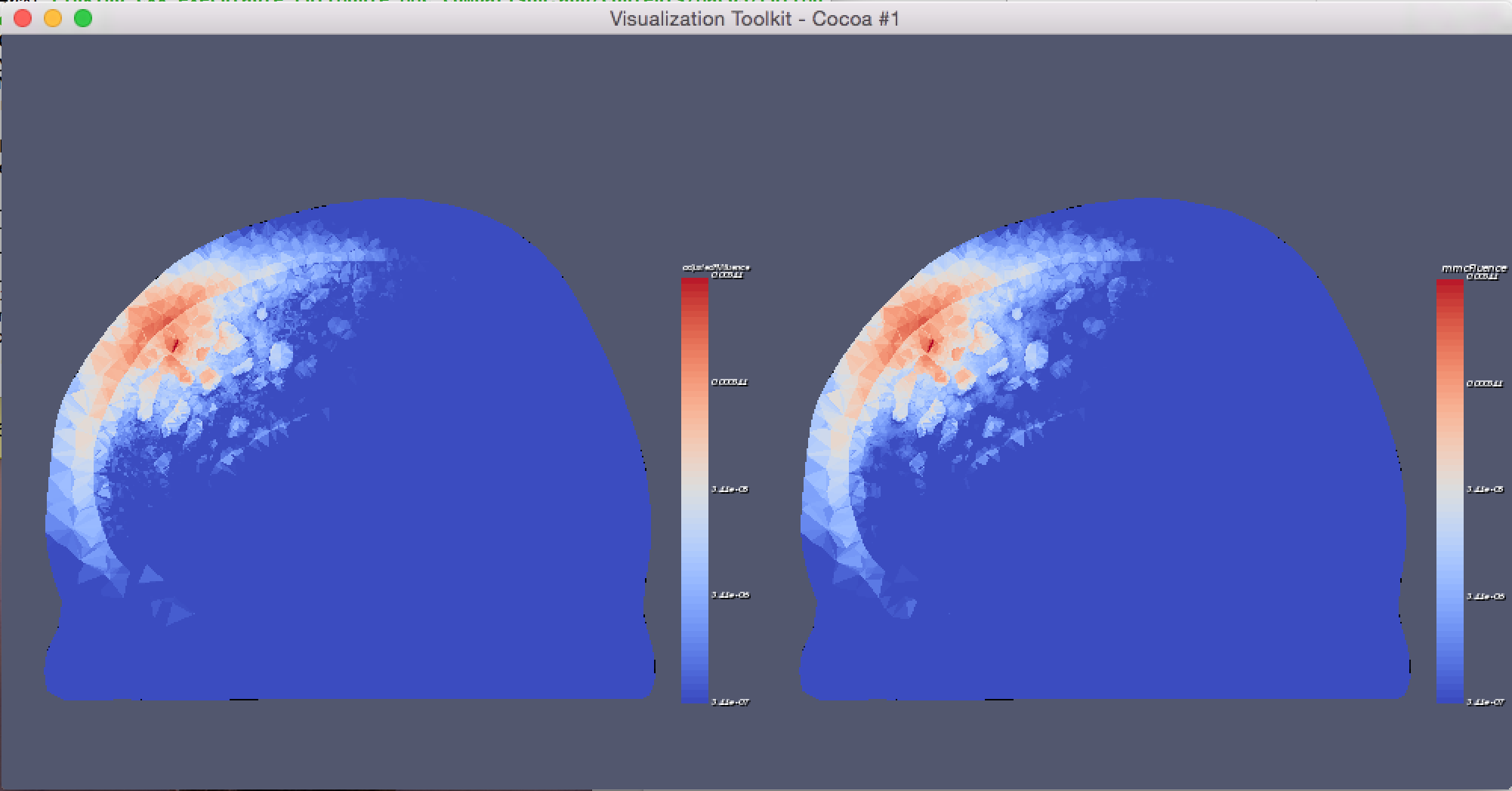
colin27 //mesh

**Step 2:** In terminal, run the pre-compiled executable:

./FullMonte\_MMC\_point FullMonte\_MMC\_point.in

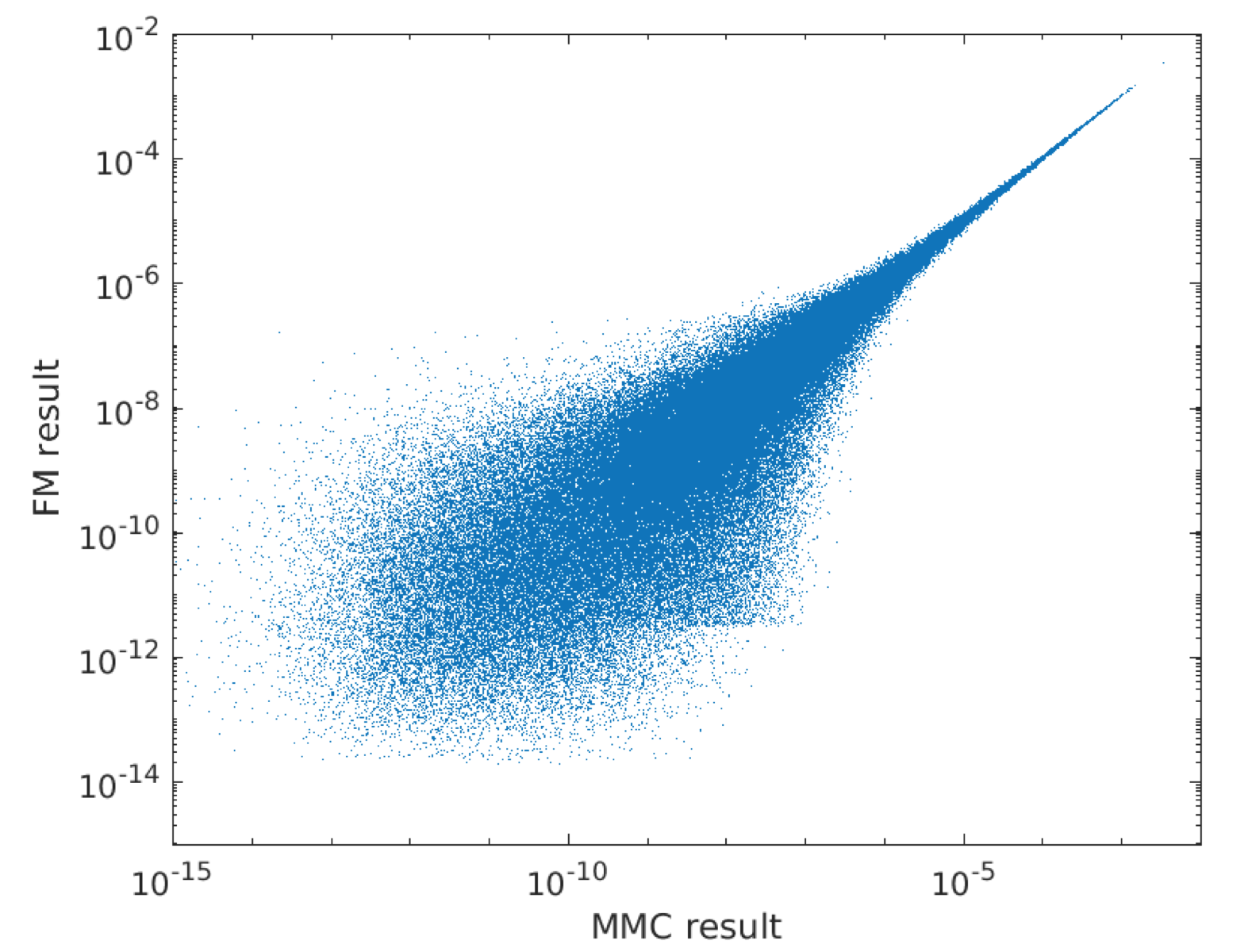
Note that to generate the data comparison plots, MatLab runtime v92 must be installed and the executable must be run in Linux-based environment. Otherwise, the plot will not be generated. Detailed information is in README.md.

**Step 3:** An interactive window should pop up. A clip of the rendered mesh model with simulation results is displayed. FullMonte energy data is normalized so that the total energy launched is equal to one, and displayed on the left. MMC is displayed on the right.



**Step 4:** In the /Results, /Results/colin27 subdirectories, the following files should be saved:

1. A 2D scatter plot (if MatLab runtime is installed). File name: comparison\_plot\_0.png



The horizontal axis represents the energy absorbed at each tetrahedron using MMC and the vertical axis is that of FullMonte. The units are in J/mm2.

1. A visualization figure in .png format. It should look similar to the window in Step 3. File name: FullMonte\_MMC\_point.png.
2. A data summary text file listing results from both simulators in order of tetrahedron id. The order in the text file is: tetraID, MMC, FullMonte.

File name: point\_energy\_data\_summary\_0.txt.

1. A text file with statistical test results, including chi-squaured statistic, p-value, absolute error between the two simulators using all data points, partial energy used for statistical analysis, and total energy recorded to be absorbed.

File name: chi2test\_results.txt

1. A .vtk file containing the mesh information, the region data, and the energy data from both simulators. The field names are: adjustedFMenergy, and mmcEnergy.

File name: point\_energy\_data\_summary\_0.vtk

Note: To open the plot and the png file from terminal, type: xdg-open fileName. This has the same effect as double-click in a file window. The text files can be opened with any text editors of your choice. The .vtk file can be used for visualization as shown in Example II.