

---

# **GSim User Guide**

***Release 1.0.0rc1***

**Tech-X Corporation**

**Dec 02, 2025**



# CONTENTS

<b>1</b>	<b>Overview</b>	<b>1</b>
1.1	What is GSimComposer?	1
1.2	GSim Capabilities	1
<b>2</b>	<b>Starting GSimComposer</b>	<b>3</b>
2.1	Running Locally	3
2.2	Running GSimComposer On a Remote Computer System	4
2.3	Visualizing Remote Data	5
2.4	Welcome Window	6
<b>3</b>	<b>Licensing</b>	<b>7</b>
3.1	Installing a License in Composer	7
3.2	Generating Licensing Logs	10
<b>4</b>	<b>Creating or Opening a Simulation</b>	<b>11</b>
4.1	Starting a Simulation	11
<b>5</b>	<b>Menus and Menu Items</b>	<b>21</b>
5.1	File Menu	21
5.2	Edit Menu	25
5.3	View Menu	27
5.4	Help Menu	27
5.5	Composer Settings/Preferences Menu	29
<b>6</b>	<b>Simulation Concepts</b>	<b>37</b>
6.1	Simulation Concepts Introduction	37
6.2	Grids	38
6.3	Geometries	42
6.4	Electric and Magnetic Fields	42
6.5	Particles	47
6.6	Reactions	48
6.7	Macroparticle Weights	50
6.8	Histories	51
<b>7</b>	<b>Advanced Simulation Topics</b>	<b>53</b>
7.1	Selecting Solvers and Solver Parameters	53
<b>8</b>	<b>Visual Setup Tab</b>	<b>55</b>
8.1	Visual-setup Simulations	55
8.2	Navigation Pane and Simulation Files	56
8.3	Elements Tree Overview	56

8.4	Description . . . . .	57
8.5	Constants . . . . .	57
8.6	Parameters . . . . .	58
8.7	Basic Settings . . . . .	59
8.8	Functions . . . . .	59
8.9	SpaceTimeFunctions . . . . .	60
8.10	Materials . . . . .	61
8.11	Geometries . . . . .	62
8.12	Grids . . . . .	72
8.13	Field Dynamics . . . . .	73
8.14	Histories . . . . .	78
8.15	Property Editor . . . . .	78
8.16	Geometry View . . . . .	78
8.17	Particle Dynamics . . . . .	81
8.18	Collisions . . . . .	109
<b>9</b>	<b>Prepare Tab</b>	<b>111</b>
<b>10</b>	<b>Run Tab (Executing the Computational Engine Vorpal)</b>	<b>113</b>
10.1	Running Vorpal within GSimComposer . . . . .	113
10.2	Running Vorpal from the Command Line . . . . .	122
<b>11</b>	<b>Output Data</b>	<b>137</b>
11.1	HDF5 Format Data Output Files . . . . .	137
11.2	Frequency for dumping simulation quantities . . . . .	137
11.3	Change the Names of Output Files . . . . .	138
11.4	Displaying the Content of .h5 Files . . . . .	138
11.5	Structure of Simulation Output .h5 Files . . . . .	139
<b>12</b>	<b>Design Tab (Parameter Scans)</b>	<b>145</b>
12.1	Introduction to Parameter Scan . . . . .	145
12.2	Using An Example To Demonstrate How To Run Parameter Scan . . . . .	145
12.3	Parameter Scan Panel . . . . .	146
12.4	Setting up the Parameter Scan . . . . .	146
12.5	Start/Stop the Scan . . . . .	150
12.6	Output pane . . . . .	150
12.7	Running an Analyzer on the Sub Simulation Data . . . . .	152
12.8	Visualization of Sub Simulation Data . . . . .	152
12.9	Directory Structure . . . . .	153
12.10	Deleting Existing Parameter Scan Data . . . . .	153
<b>13</b>	<b>Data Analysis Tab</b>	<b>157</b>
13.1	Overview of Using Analyzers . . . . .	157
<b>14</b>	<b>Visualization Tab</b>	<b>161</b>
14.1	Introduction to the Visualize Window . . . . .	161
14.2	Select the Visualize Icon from the Icon Panel . . . . .	161
14.3	Data View Pull-down Menu . . . . .	161
14.4	Standard Controls Available Across Multiple Views . . . . .	162
14.5	Data Overview . . . . .	168
14.6	Field Analysis . . . . .	173
14.7	History . . . . .	176
14.8	Phase Space . . . . .	179
14.9	Binning . . . . .	181



<b>15</b>	<b>Troubleshooting</b>	<b>185</b>
15.1	Troubleshooting Electrostatic Simulations . . . . .	185
15.2	Troubleshooting Electromagnetic Simulations . . . . .	185
15.3	Troubleshooting Visual Setup Crashes . . . . .	186
15.4	Troubleshooting Plasma Density . . . . .	190
15.5	Troubleshooting Missing Secondary Particles . . . . .	190
15.6	Troubleshooting Crashes During Stepping of Particle Simulations . . . . .	190
15.7	Troubleshooting Performance . . . . .	191
15.8	Troubleshooting MPI failure to start on OSX . . . . .	192
15.9	Troubleshooting Windows Permissions . . . . .	193
15.10	Troubleshooting GSimComposer Visualization . . . . .	193
<b>16</b>	<b>Glossary</b>	<b>195</b>



## OVERVIEW

This manual demonstrates how to use either the Visual Setup (.sdf input files) or Text Base Setup (.pre input files) to set up a GSim simulation. It then shows how to run the computational engine on the resulting input files, how to perform data analysis in GSim, and how to visualize data.

GSim is an arbitrary dimensional, electromagnetics and plasma simulation code consisting of two major components:

- GSimComposer, the graphical user interface.
- Vorpall [NC04], the GSim Computational Engine.

GSim also includes many more items such as Python, MPI, data analyzers, and a set of input simplifying macros.

### 1.1 What is GSimComposer?

GSimComposer provides an interface that allows you to edit and validate your simulation input files, run GSim simulations, and visualize results using the VisIt-based Visualization pane embedded within GSimComposer. You can also edit GSim input files in the text editor of your choice, perform calculations with the easy-to-run, command-line-driven Vorpall engine, and then run the visualization tool of your choice.

---

**Note:** Look and Feel Differences

GSim runs on Linux, Mac OS X, and Windows. Each of these platforms has its own unique look and feel (e.g. the ordering of buttons in dialogs, and in the case of Mac OS X the arrangement of menus and menu items). Furthermore, the appearance of GSim can vary depending on the theme being used.

Given these differences and that screenshots may not always be of the most recent release, please note that screenshots shown in this manual may look different from the GSim that you see running on your own computers. GSim should function the same, but if you look for a particular toolbar or button, it may not be in same place in your copy of GSim.

---

### 1.2 GSim Capabilities

GSim is a flexible, multiplatform, high-performance tool for running electromagnetic, electrostatic, and plasma simulations in 1, 2, or 3 dimensions.

The structures within the GSim applications can be arbitrarily shaped and can define the locations of conductors, dielectrics, particle absorbers, reflectors, and other geometrical objects.

GSim solves EM propagation in the presence of complex dielectric and metallic shapes with accurate simulation of curved geometries using a conformal mesh. Examples include:

- Dish Antenna

- Horn Antenna
- Patch Antenna
- Waveguides
- Far Field calculations
- Radar Cross Sections
- Crab Cavities

## STARTING GSIMCOMPOSER

### 2.1 Running Locally

Once you have installed GSim successfully as per the instructions given by “Installation section”, you are ready to run the program. This section will detail how to run GSim locally on Windows, Mac, and Linux operating systems.

#### 2.1.1 Running GSim on Windows

You can start GSim in the following ways on Windows:

- Select the GSim Icon
  - Double-click on the GSim shortcut on your desktop
  - Go to your Start menu, navigate to the Tech-X folder, and click on the GSim icon
- Run GSim from the Windows Command Line
  - Navigate to the relevant program folder by going to *AppData -> Roaming -> Tech-X -> GSim-1.0 -> Contents -> bin*
  - Type in GSimComposer.exe to run the GSim executable and launch the program
- Open GSim through Windows File Explorer
  - Open Windows File Explorer, either through your start menu, desktop shortcut, or taskbar icon
  - Navigate to *C:\Users\%USERNAME%\AppData\Roaming\Tech-X\GSim-1.0*
    - \* You can either click on the GSimComposer.lnk shortcut in this folder, or...
    - \* Continue on to *Contents -> bin* and then double-click on GSimComposer.exe
- Open GSim using an existing file
  - Open Windows File Explorer
  - Select a file that has a GSim-supported extension (like a .pre, .in, or .sdf, for example). Either...
    - \* Double-click on the file if its default program is GSimComposer
    - \* If its default program is not GSimComposer, right-click on the file and select **Open with**. Go into *AppData -> Roaming -> Tech-X -> GSim-1.0 -> Contents -> bin* and select the GSimComposer.exe application file.

---

**Note:** There is a known Windows-OpenGL issue when using GSim over a Microsoft Remote Desktop connection where the application window appears blank on some Windows machines with some graphics cards or some drivers. We provide a RemoteGSimComposer.bat script to work around this issue. It disconnects the remote session and starts

---

GSim while disconnected, so that when the user reconnects, GSim works properly. If you see this issue, launch this script by right-clicking on RemoteGSimComposer.bat and choose “Run as administrator”.

---

## 2.1.2 Running GSim on Mac

On Mac OS, the methods for starting GSim are:

- Start from the Applications Folder
  - Click on the GSimComposer icon in the Applications/GSim folder (or in the folder where you have GSim installed)
- Run GSim from Terminal Window
  - Open a Terminal window
  - Navigate to the folder where GSimComposer is installed, most likely by going to /Applications/GSim-1.0/GSimComposer.app/Contents/MacOS
  - Start GSim by typing ./GSimComposer.sh to run the program executable

## 2.1.3 Running GSim on Linux

For Linux, you can start GSim through the following:

- Navigate to the folder where the program is installed, for example /user/local/GSim-1.0-Linux64
- Type in ./GSimComposer.sh to run the program executable

## 2.2 Running GSimComposer On a Remote Computer System

Just as on a local workstation or laptop, the computational engine (Vorpal) may be invoked through the graphical interface or from the command line on a remote system. On high performance computing clusters the command line approach may be required in order to submit a job to a resource management system. These are documented separately here: - [Running Vorpal from the Command Line](#) - [Running Vorpal from a Queueing System](#)

In this section we discuss alternatives for setting up, running via the GUI, and visualising output.

In the present version we offer the following capabilities for running GSim remotely:

---

**Note:** Prior to starting up GSim, it may be necessary to set the environment variable `export LIBGL_ALWAYS_INDIRECT=1` in order for the visualization stage to work correctly.

---

---

**Note:** Some users using VNC on ubuntu 14.04 have also reported adding the system installation of mesa to the start of the `LD_LIBRARY_PATH` environment variable, has helped overcome their issues. `export LD_LIBRARY_PATH=/usr/lib/x86_64-linux-gnu/mesa:$LD_LIBRARY_PATH`

---

- VNC software (e.g., Tight VNC or Turbo VNC): If a VNC server is set up on the remote machine, one may connect a local client to the remote machine using VNC.
- NoMachine: In order to log in to a remote server, the same software needs to be installed on both the server and the client. Please work with your local support staff to ensure any one of these software solutions is correctly installed on both the server and client.

- Virtual GL: If one is using a remote machine that has virtualGL server or DCV one may run using the hardware acceleration on the remote machine. This may provide the best performance but also the most system administration work. Many HPC centers are already set up for this kind of access. There is a super-accelerated virtualGL client, but it is more common to find virtualGL set up like DCV such that the remote machine is running a VNC server, which you may connect to with any VNC client on your local machine.
- X Windows: If one does not have hardware acceleration on the remote machine one may forward X using an ssh client (`ssh -Y`) or use accelerated X forwarding using software like NoMachine NX. As of this writing, a good discussion is at [https://www.hoffman2.idre.ucla.edu/access/x11\\_forwarding](https://www.hoffman2.idre.ucla.edu/access/x11_forwarding). Briefly,
  - Linux users running X: edit `/usr/bin/Xorg` as described at the above link.
  - Windows users: many options described at the above link. In addition, you can install an X server (such as Xming) on your Windows machine. You will also need to install Cygwin. Follow these steps to open Composer on a remote Linux machine. (1) Start the Xming server on your local computer. (2) Open Cygwin and type the following commands in the Cygwin terminal:
    - \* `startxwin &`
    - \* `export DISPLAY=:0.0`
    - \* `xterm &`
    - \* `ssh -XY address.to.your.remote.machine`
    - \* navigate to location of installed GSim software and issue the command `./GSimComposer.sh`

If X11 forwarding is enabled on the remote machine, then this should open a Composer window.
- Microsoft Remote Desktop: There can be problems due to OpenGL issues, in which case one can use the RemoteDesktopComposer.bat script provided with GSim.

## 2.3 Visualizing Remote Data

The following capabilities are recommended for visualizing remote data if the previous recommendations do not work for you:

- One may use an external utility to copy the remote files back to a local machine (`scp`, `sftp` or `winSCP` are likely options). Providing the `.pre` or `.sdf` file is in the same directory as the data to be viewed, it should be possible to visualize the output locally. This is the most appropriate solution for those dealing with small datasets.
- Use standalone VisIt and its remoting feature. VisIt may be downloaded from (<https://wci.llnl.gov/simulation/computer-codes/visit/downloads>). Users of native VisIt should be sure to make sure their local version number matches up with the version running remotely. This option offers the full flexibility of VisIt, such as the ability to make all the individual images needed to make movies in a single go and the ability to fully Python script your visualisation, but there is a corresponding learning curve. There are many tutorials and resources at The visitusers website (<http://www.visitusers.org>).
- Use matplotlib or alternative software on the remote machine. The VsHdf5 module is provided with GSim and may be used to read and manipulate remote datasets for this purpose.
- If you are using VNC or NoMachine, you can open any data set through Composer just as you would on your own desktop.

## 2.4 Welcome Window

Upon opening GSim, you are brought to the Welcome Window. If it is your first time opening GSim, your *Recent Simulations* will be empty. However, if you have completed previous runs, you may use this area to quickly select a recent simulation and re-open it. You can also create new simulations based on those you have recently worked with. The Welcome Window is shown in *Welcome Window*.



Fig. 2.1: Welcome Window



## LICENSING

### 3.1 Installing a License in Composer

In order to use most features in Composer, you must have a valid license file.

#### 3.1.1 Tech-X License File (.txlic)

When you request an evaluation or purchase a license for a Tech-X product, a member of the Tech-X Sales Team will provide you with a Tech-X License File which terminates in a .txlic file extension. The license file contains a unique signature that will be corrupted if any information in the file is changed. The loaded license file must be present at all times while in use.

#### 3.1.2 License Installation Locations

The license manager uses a hierarchy to look for active license files and will use the first valid license that it finds.

##### License Search Hierarchy

1. *Environment Variable:* The first place that the license manager looks is to check if the <PRODUCT>\_LICENSE\_DIR environment variable (e.g. VSIM\_LICENSE\_DIR or RSIM\_LICENSE\_DIR) is defined and exists. To use this method to set the active license, two conditions must be met:
  - (1) The environment variable must be set to the full path of the *directory* containing the license file (not the path the .txlic file itself)
  - (2) the license file name must be exactly named <product-lowercase><major-version>.txlic (e.g. vsim12.txlic or rsim4.txlic). See [Setting the License Directory Environment Variable](#) for help setting the environment variable.
2. *User Directory:* If the environment variable is not set, the license manager will proceed to look in the User Directory for the license file. See the table below for examples of the path which varies per operating system.

Table 3.1: Example paths to Documents directory for GSim

Operating System	User License Directory
MacOS/Linux	/Users/<username>/Documents/Tech-X/licenses/GSim
Windows	C:\Users\<username>\Documents\Tech-X\licenses\GSim

3. *Common Directory*: Finally, the license manager will look in a common directory. The purpose of this location is the ability to share a license by multiple users using the same computer. See the table below for examples of the path which varies per operating system.

Table 3.2: Example paths to the Common Directory for GSim

Operating System	Common Directory
MacOS/Linux	/Users/Shared/Tech-X/licenses/GSim
Windows	C:\ProgramData\Tech-X\licenses\ GSim

### 3.1.3 Managing Licenses within Composer

The easiest way to install a license for use with this product is to use Composer's user interface for installation.

1. Open Composer.
2. Open Preferences (MacOS) or Settings (Windows/Linux).
3. Navigate to the License Settings window.
4. Select the Load button for either the User Directory (one user) or the Common Directory (multiple users).
5. Select the license file to use from the file selection dialog, and load the file. Make sure the status is VALID.
6. Click OK to close the Settings menu and proceed with your simulation.

### 3.1.4 Setting the License Directory Environment Variable

Using an environment variable allows you to save your license file to any directory you'd like and use it from there. Each Tech-X product has a unique environment variable. Reference the table below to find the relevant variable name.

1. Determine the appropriate environment variable name for your product from the table below:

Table 3.3: Environment Variable names by product

Product Name	Environment Variable Name
VSim	VSIM_LICENSE_DIR
XSim	XSIM_LICENSE_DIR
RSim	RSIM_LICENSE_DIR
PSim	PSIM_LICENSE_DIR

- Note: The value of the environment variable must be set to the **directory** containing the file, not the file itself.
2. Name your license file. The format should be all lower case in the form <product-name><major-version>.txlic. If your license file does not follow this convention, it will not be found.  
*e.g. vsim12.txlic or xsim1.txlic*
  3. Once you have your license file named correctly in the desired directory, set the variable in the system environment:
    - a. On Mac/Linux, open a terminal to set the environment variable:

```
export VSIM_LICENSE_DIR=/path/to/license/directory/
```

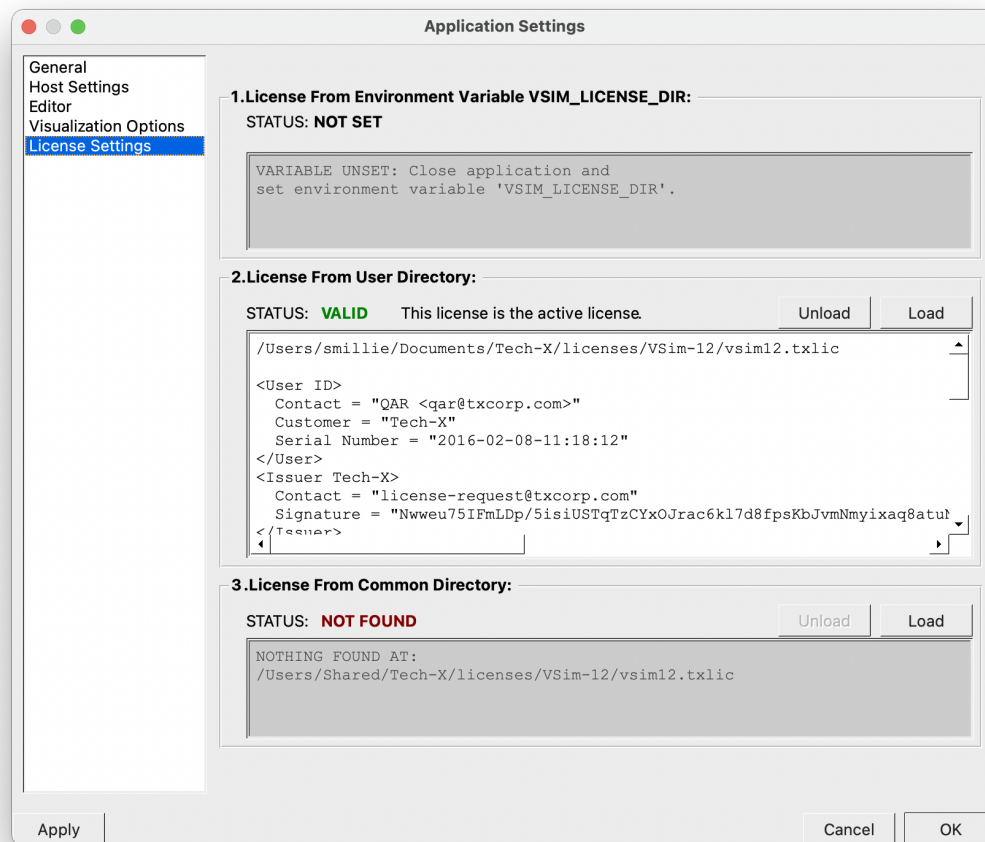


Fig. 3.1: Managing Licenses in Composer

- b. On Windows 10+, click the Windows icon, type “environment” and select “Edit the system environment variables” and follow the instructions on that dialog to set the variable to the correct path.
4. Start composer with the same environment.

## 3.2 Generating Licensing Logs

If you experience any license-related issues with this product, Tech-X support may request logs generated by the license manager. Logging is disabled by default, so it will need to be enabled in order to generate these logs. Follow these steps to enable logging:

1. Close Composer if it was previously running.
2. Create a new directory which the logs will be saved to. This can be any directory as long as composer has access to write to it.
3. In your newly created log directory, you will need to create (touch) empty files for the license manager to write to. These files should be named in the following convention: `<executableName>License<mpiRank>.log`. For serial runs, the mpiRank should be 0. For parallel runs, the range of possible log files is 0 to number\_of\_cores minus 1. For example, if you are running Vorpai with 4 cores, to get the Vorpai license log files for the first (mpi rank 0) and last (mpi rank 3) mpi process, create both `vorpaiLicense0.log` and `vorpaiLicense3.log` files.
4. Set the environment variable **TECHX\_LOG\_DIR** to the log directory.
  - a. On Mac/Linux, this can be done as following:

```
export TECHX_LOG_DIR=/path/to/log/directory/
```

- b. On Windows 10+, click the Windows icon, type “environment” and select “Edit the system environment variables” and follow the instructions on that dialog to set **TECHX\_LOG\_DIR**
5. Open Composer and reproduce the issue. Logs will be generated as you work.

## CREATING OR OPENING A SIMULATION

### 4.1 Starting a Simulation

The first step to creating your own simulation is to open a file. It can be an existing example, an existing simulation, or a completely new simulation.

#### 4.1.1 Creating a Simulation from an Example

To run one of the examples in GSim, one must first copy the example file, and any correlated files, to a new directory. Choose *New -> From Example* from the **File** menu. See *Create new simulation from example*.

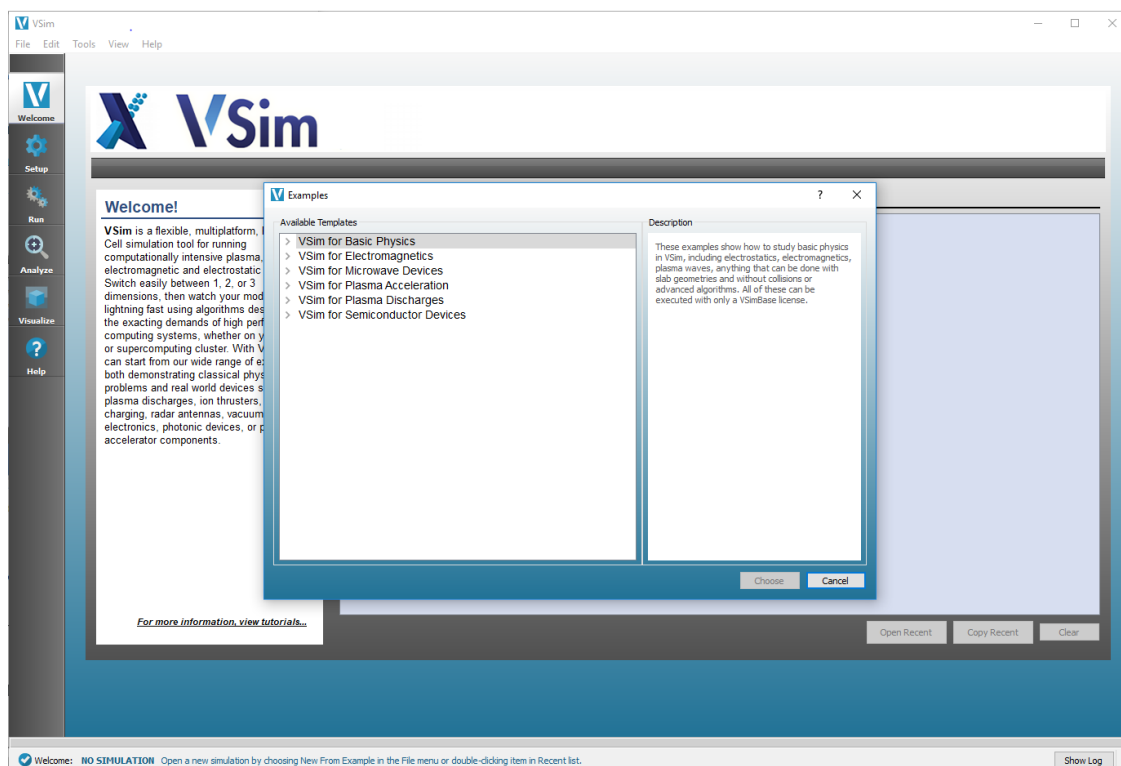


Fig. 4.1: Create new simulation from example

## Select Example Template

The *Examples* dialog box will open and you can choose an example template to run. Here a short description and image of the available examples will be displayed. Examples are split into *GSim Modules*. You must possess the correct license for an example to run, though you can see them all.

Upon selecting an example from the *Available Templates* pane, either double click the example you have chosen or single click and then click the *Choose* button.

For this demonstration we have chosen the *Electromagnetic Plane Wave* example from the *GSim for Basic Physics* module. See *Selecting an example from the Examples dialog*.

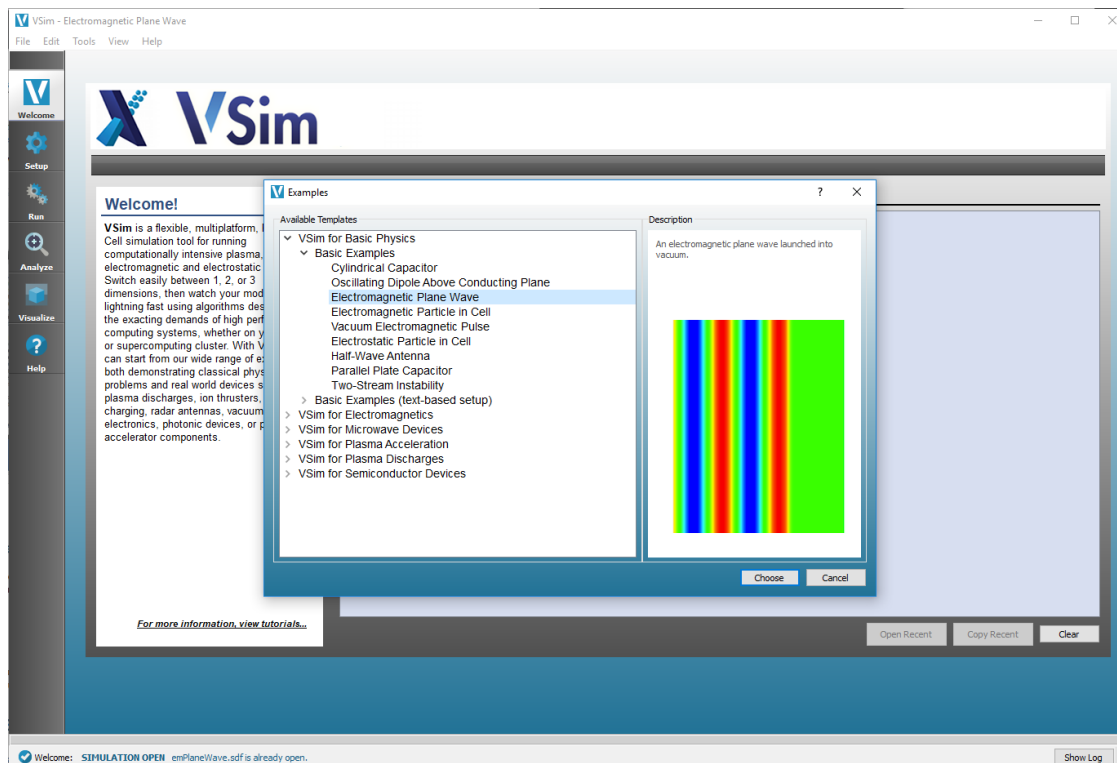


Fig. 4.2: Selecting an example from the *Examples* dialog

## Choose a Name for the New Simulation

Upon selection of the example template, the window *Choose a Simulation Name* will open. Here you can choose the folder, or create a new folder, and set the name of the new simulation.

When the name and directory have both been chosen, click the *Save* button to proceed. See *Choose name and directory to use for the simulation*.

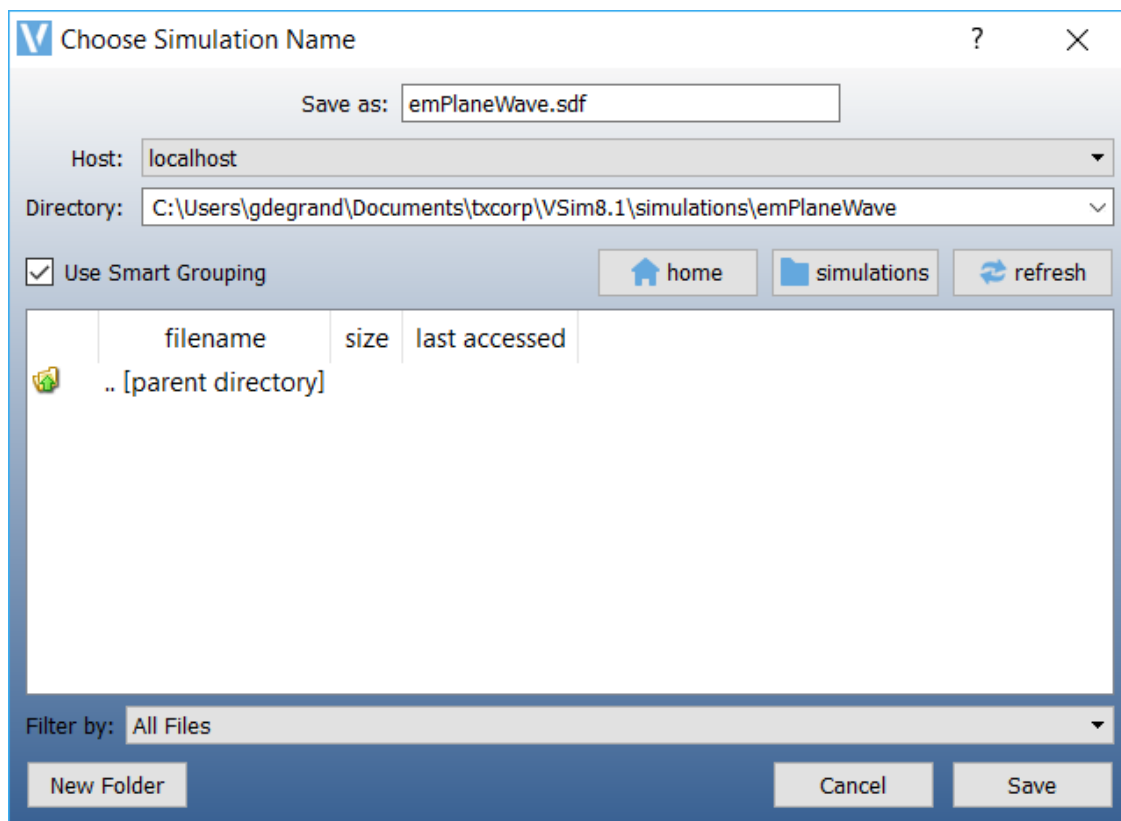


Fig. 4.3: Choose name and directory to use for the simulation

## Proceed to the Setup Window

From here, Composer will automatically take you to the **Setup** window where you are free to explore and alter the parameters of the example you have chosen. See [Proceed to the setup window](#).

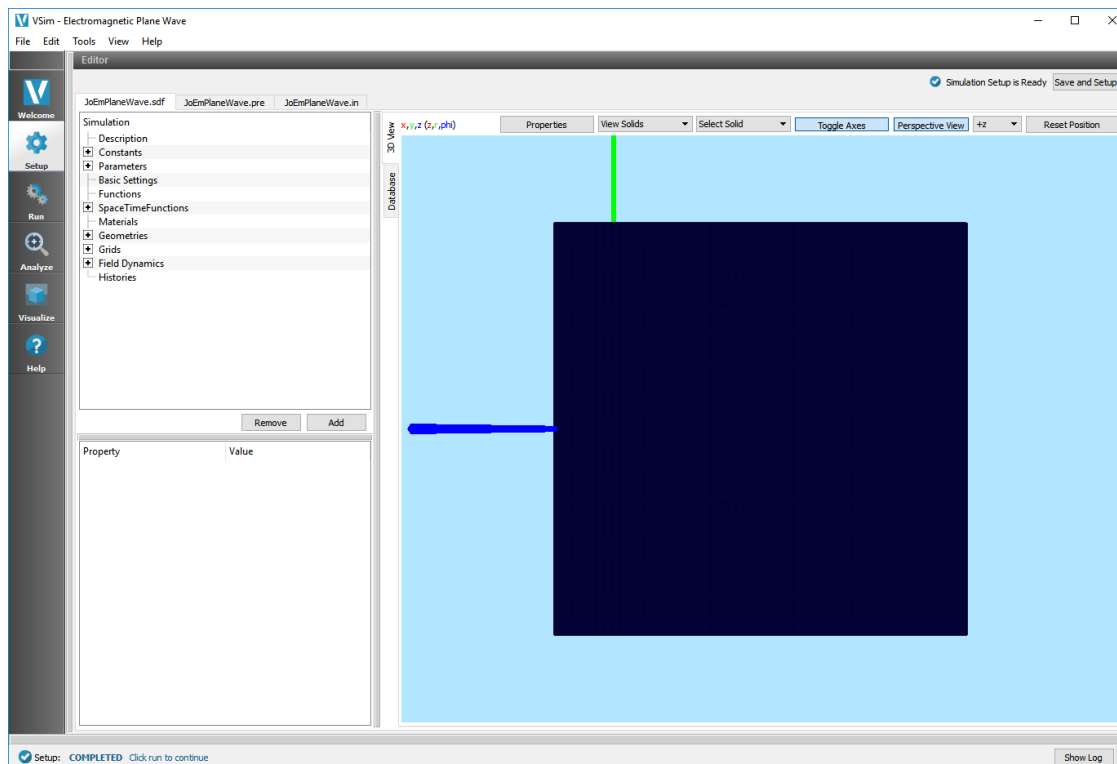


Fig. 4.4: Proceed to the setup window

### 4.1.2 Opening an Existing Simulation

If you have a simulation already created, you can double-click on the simulation .sdf or .pre file that appears under *Recent Simulations* in the GSimComposer **Welcome** menu.

Or, if your simulation does not show under the *Recent Simulations* heading, you can go to *File -> Open Simulation* and use the *Open Simulation* file navigation window to select the simulation .sdf or .pre file you would like to open. See [Open an existing simulation .sdf or .pre file](#).

From here, Composer will automatically take you to the **Setup** window where you are free to explore and alter the parameters of your simulation.



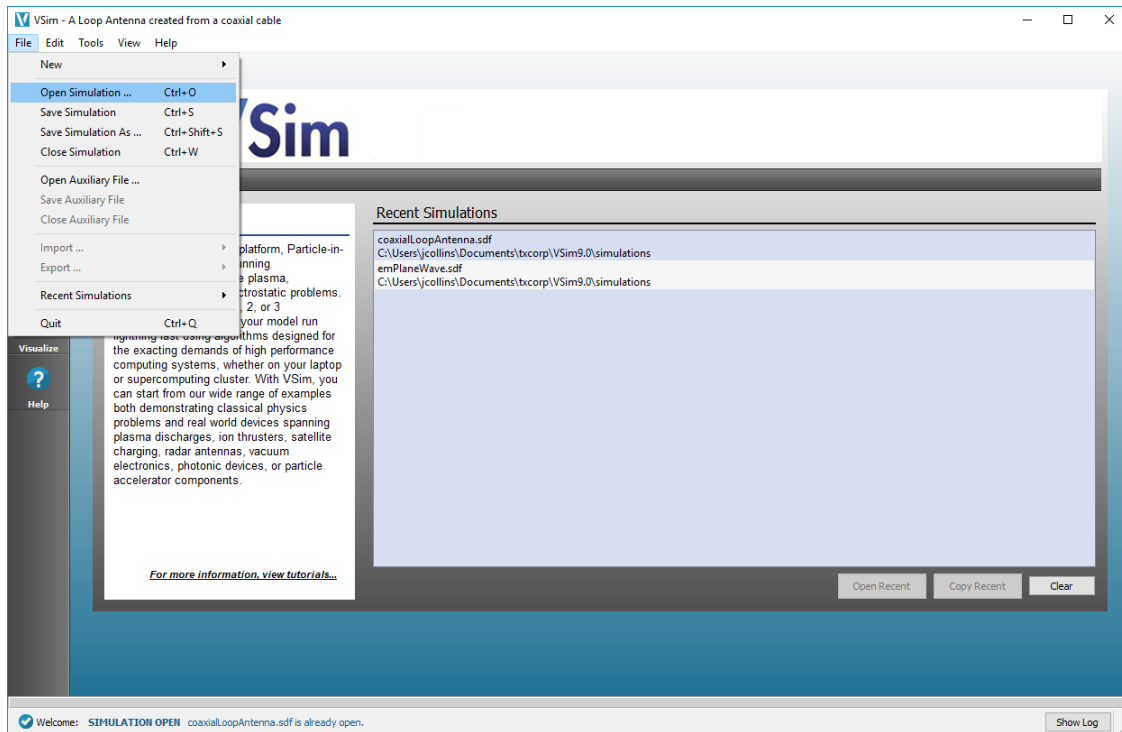


Fig. 4.5: Open an existing simulation .sdf or .pre file

### 4.1.3 Creating a Blank Simulation

#### Create New Visual Setup Simulation

To create a completely custom visual-based simulation, you must choose *New -> Simulation* from the **File** menu. Creating an entirely new simulation requires knowledge of topics covered in later sections. See [Choose to create a new visual setup simulation](#).

#### Proceed to the Setup Window

You are automatically taken to the **Setup** window. You are now free to create your own custom simulation and explore the power and versatility of the Vorpel engine. See [A blank slate](#).

#### Choose Simulation Name and Save

When you are ready to save your simulation, go to *File -> Save Simulation* and choose a location and name for your new simulation. See [Choose a name and directory for your new visual setup simulation](#).

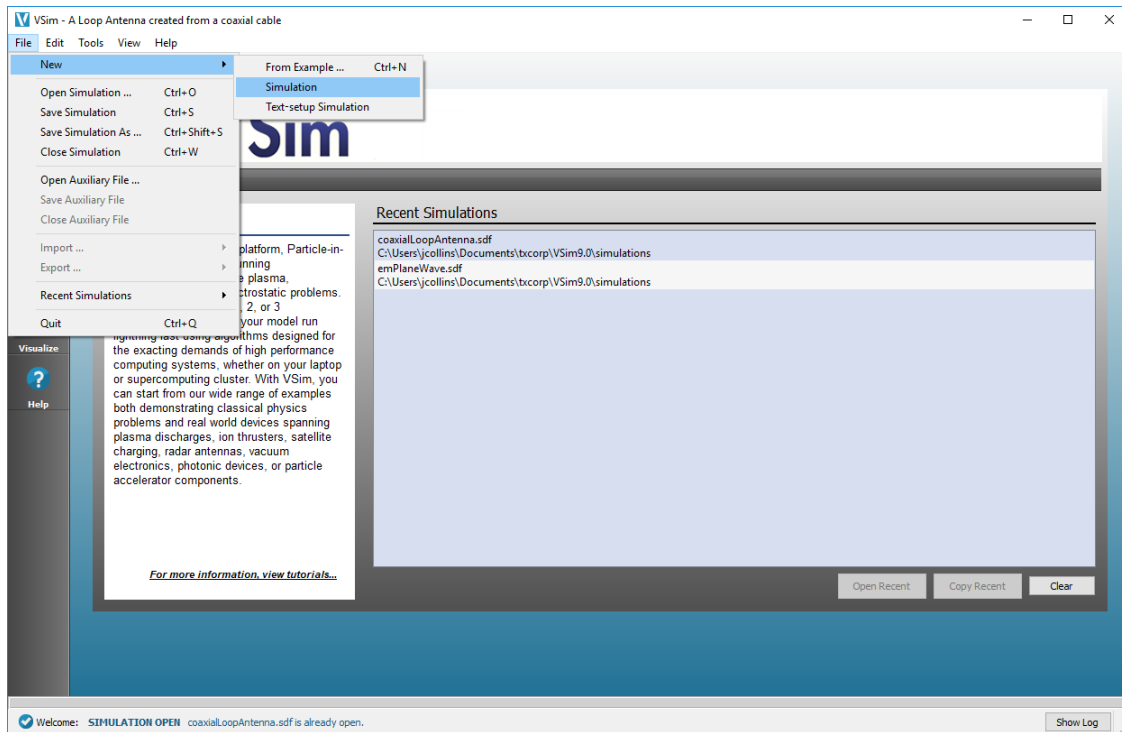


Fig. 4.6: Choose to create a new visual setup simulation

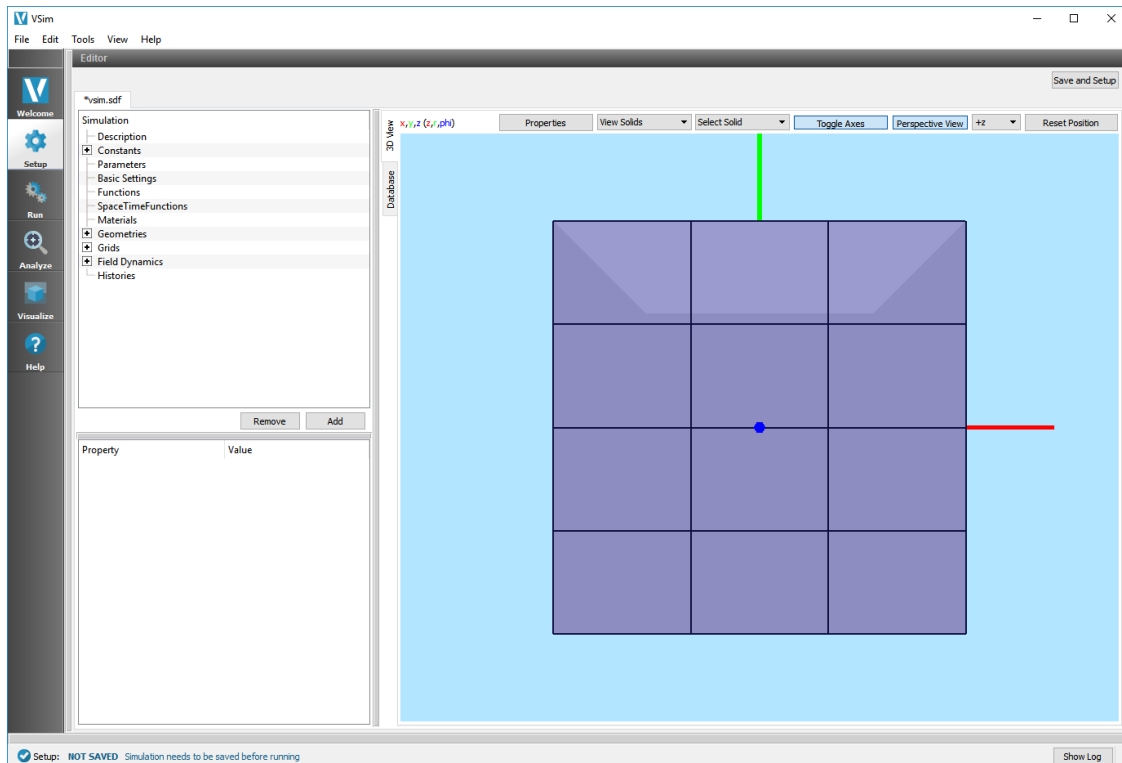


Fig. 4.7: A blank slate

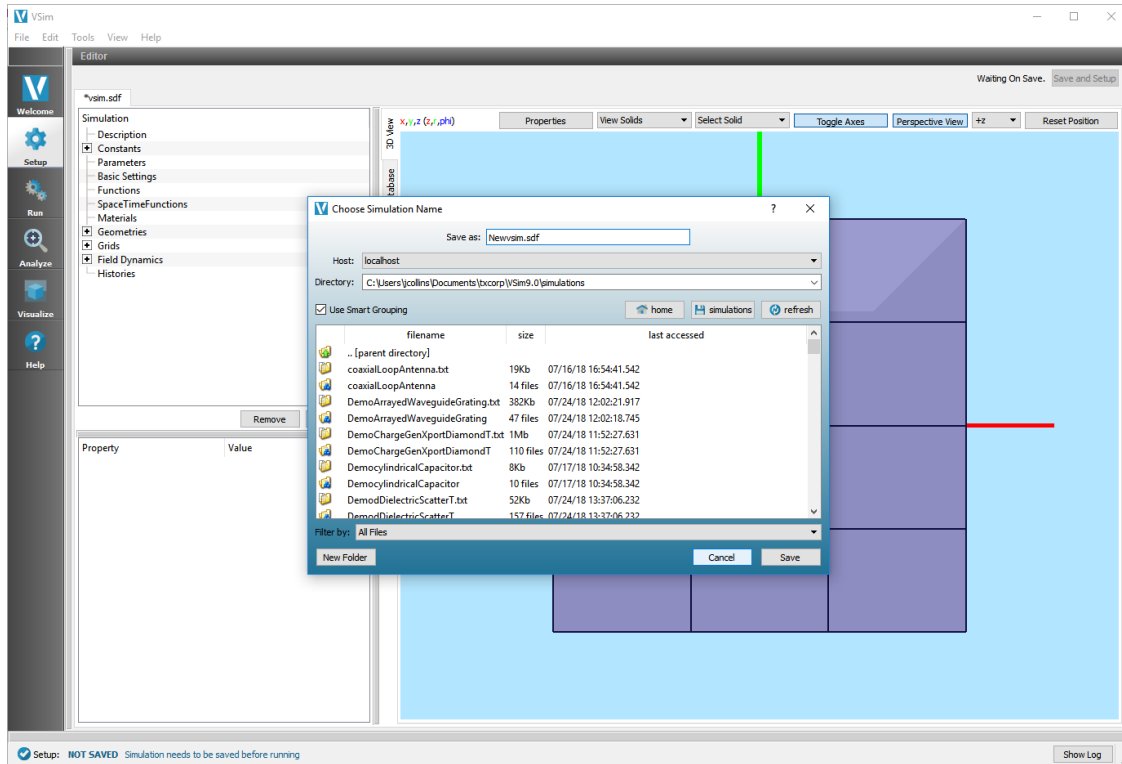


Fig. 4.8: Choose a name and directory for your new visual setup simulation

#### 4.1.4 Create New Text Setup Simulation

To create a completely custom text based simulation, you must choose *New -> Text-setup Simulation* from the **File** menu. Creating an entirely new simulation requires knowledge of topics covered in later sections. See [Choose to create a new text setup simulation](#).

#### Choose New Simulation Name And Save

A new window entitled *Choose a name for the new simulation* will pop up. Here, you choose a name for your new simulation along with a location where the simulation will be saved. See [Choose a name and directory for your new text simulation](#).

#### Proceed to the Setup Window

You are automatically taken to the **Setup** window. You are now free to create your own custom simulation and explore the power and versatility of the Vorpel engine. See [A blank slate](#).

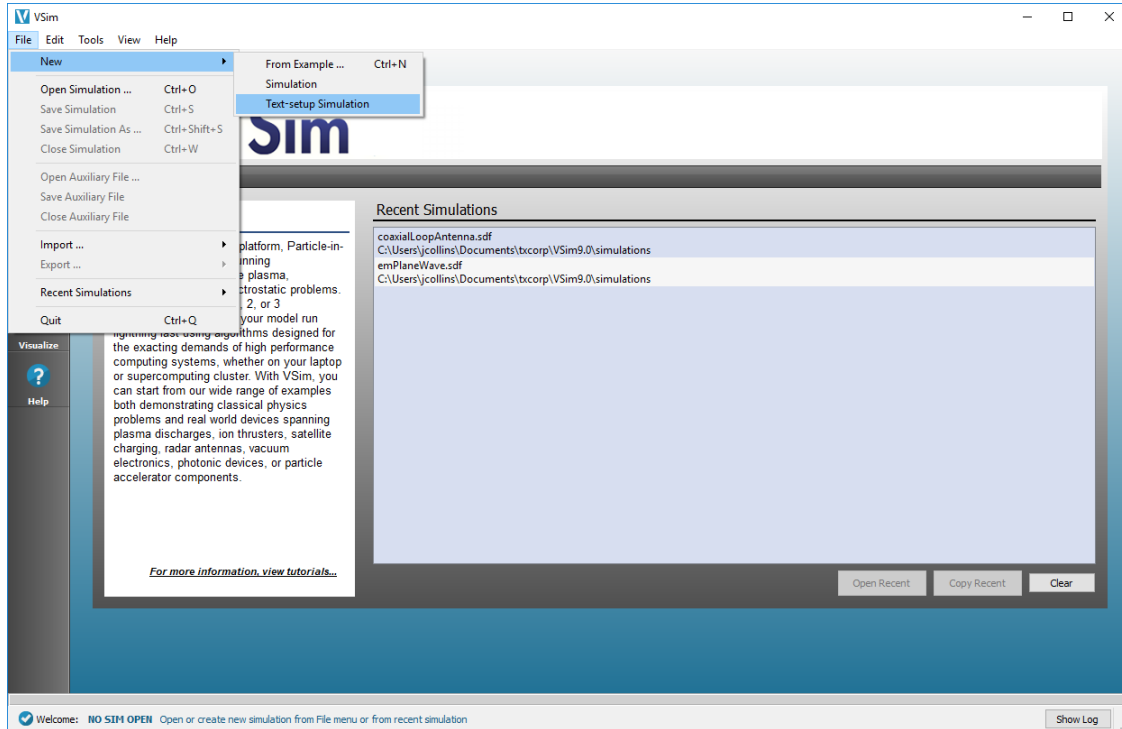


Fig. 4.9: Choose to create a new text setup simulation

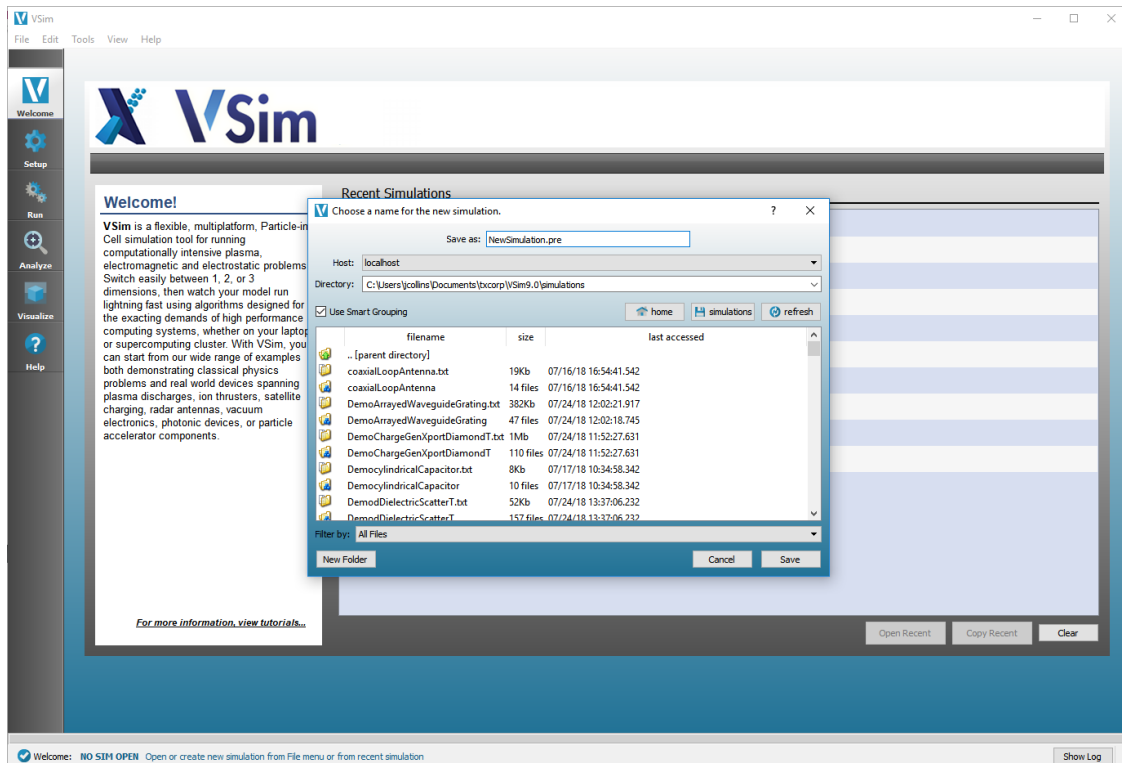


Fig. 4.10: Choose a name and directory for your new text simulation

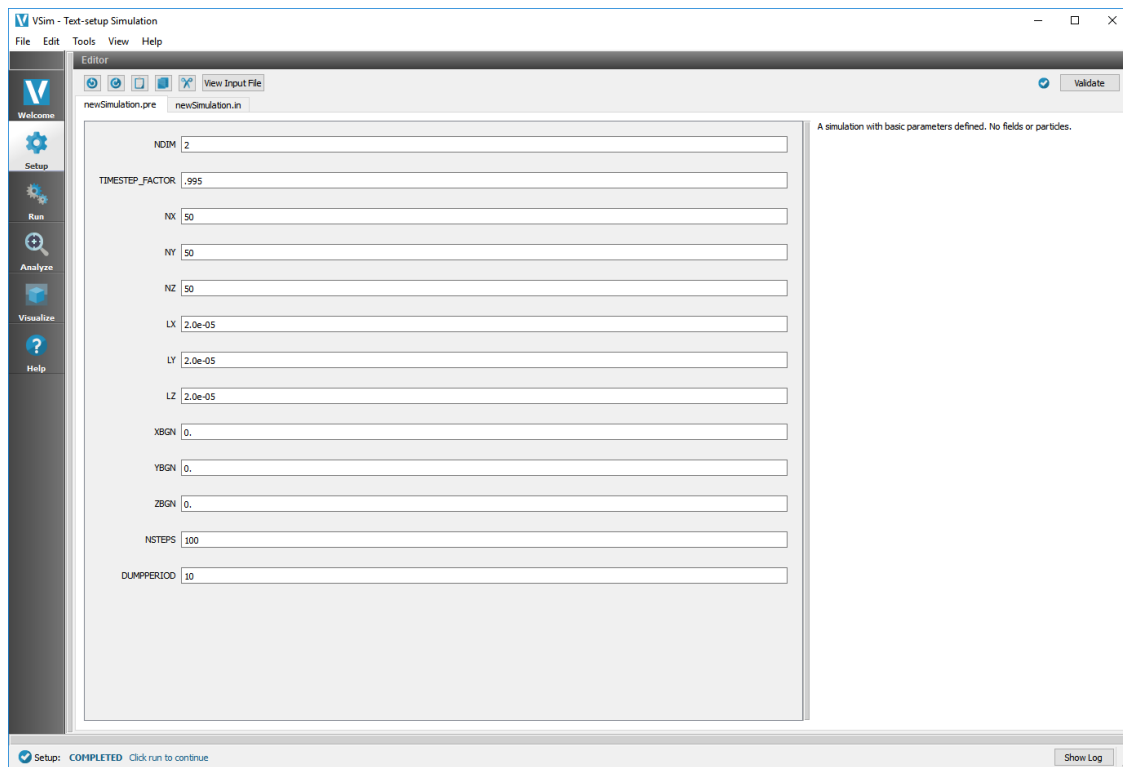


Fig. 4.11: A blank slate



## MENUS AND MENU ITEMS

### 5.1 File Menu

The **File** menu contains options to control creating, opening, closing, and saving GSimComposer files and simulation directories. See *File Menu*.

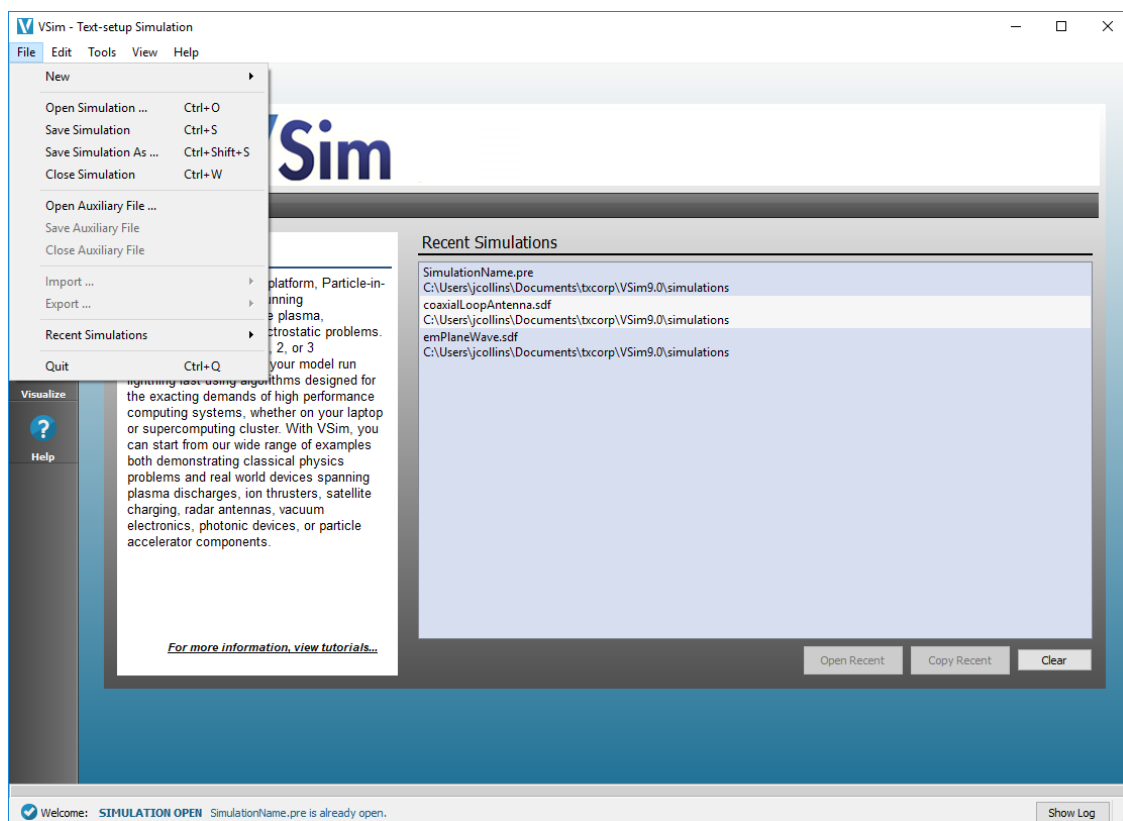


Fig. 5.1: File Menu

## 5.1.1 New

*File* -> *New* has three options:

- From Example
- Simulation
- Text-setup Simulation

For more information on each of these options, please see *Creating or Opening a Simulation*

## 5.1.2 Open Simulation

To open a directory where existing simulation files reside, select *Open Simulation* from the **File** menu. See *Open selection from File menu*.

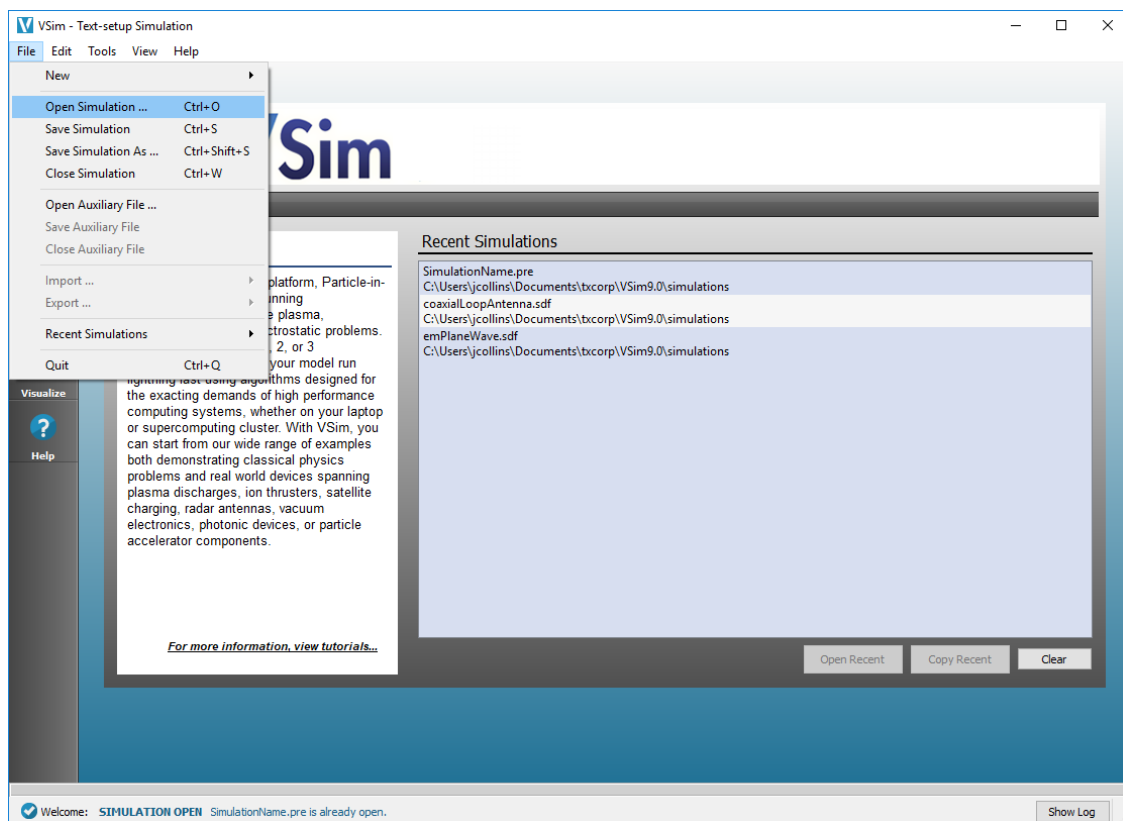


Fig. 5.2: Open selection from File menu

The default directory, **simulations**, is created for you during installation of GSim. You can modify this in the *Tools* -> *Settings* -> *Host settings* -> *Paths* menu. See *Open Simulation window*.



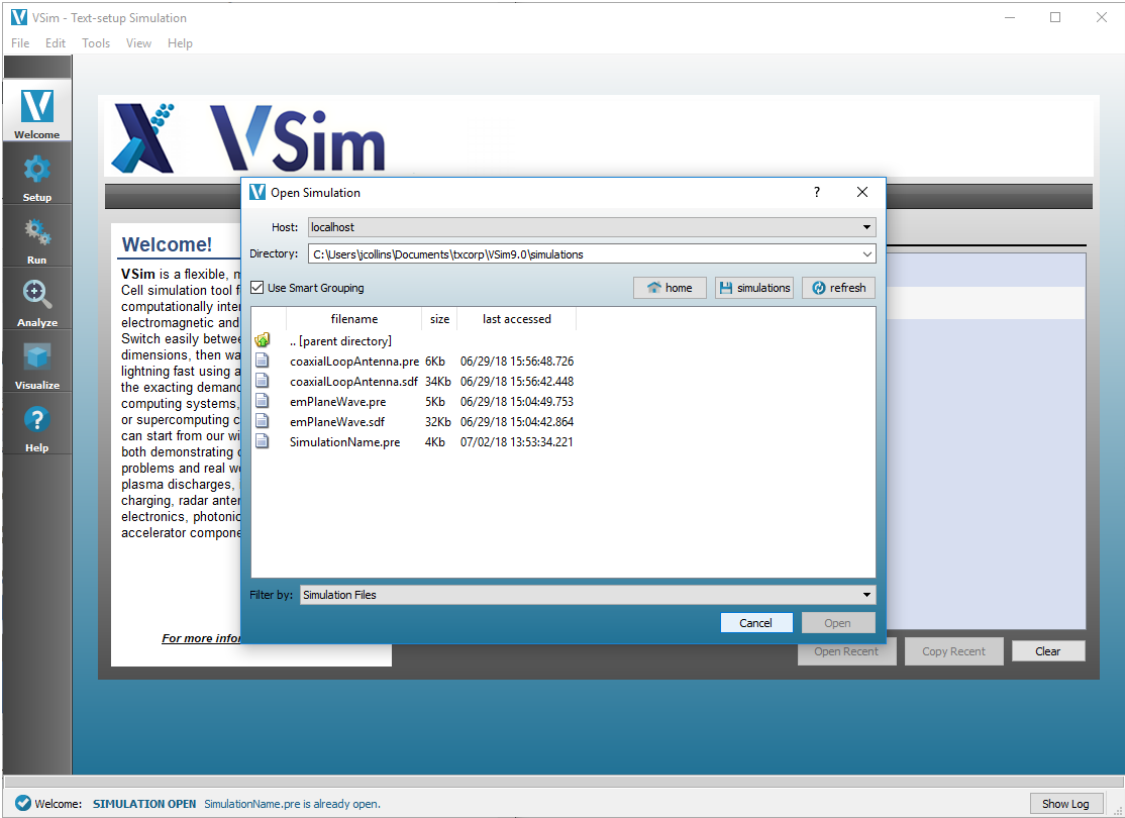


Fig. 5.3: Open Simulation window

### 5.1.3 Save Simulation

After a simulation is open, going to *File -> Save Simulation* allows you to save any modifications to the input file. The saving process will also run the Python pre-processor on the file in order to do any Python substitutions or calculations and to create the .in file.

### 5.1.4 Save Simulation As

After a simulation is open, clicking on *File -> Save Simulation As* allows you to save any modifications to the input file and write it in to a new file. The saving process will also run the Python pre-processor on the file in order to do any Python substitutions or calculations and to create the .in file.

### 5.1.5 Close Simulation

After a simulation is open, *File -> Close Simulation* allows you to close the current simulation and will return you to the **Welcome** window.

### 5.1.6 Open Auxiliary File

This can be used to open any file related to a simulation. It is most useful for opening and editing Python files. Many times, a simulation will have an associated Python file that contains script for post-processing in the **Analyze** window or for importing an external magnetic field.

### 5.1.7 Save Auxiliary File

This can be used to save any changes made to an auxiliary file.

### 5.1.8 Close Auxiliary File

This can be used to close an associated auxiliary file that was previously opened.

### 5.1.9 Import

When doing a visual-setup simulation, use the *File -> Import* option to import materials or geometries to use in your simulation.

Materials files must be of the extension “.vmat”

Geometry files can be any of .stl, .ply, .gds, .vtk, .stp, .step or .p12.

### 5.1.10 Recent Simulations

To access data to visualize from a recently conducted simulation, select *Recent Simulations* from the **File** menu in the menu bar. See *Recent Simulation Selection in File Menu*.

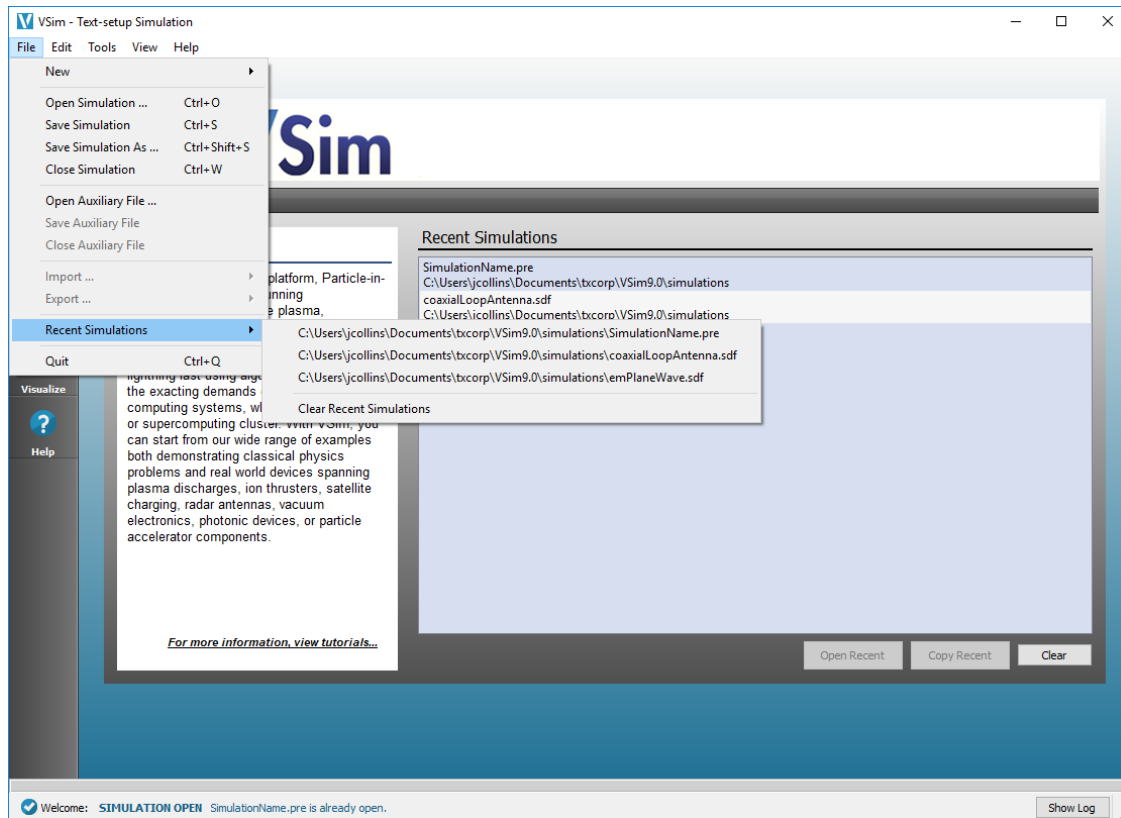


Fig. 5.4: Recent Simulation Selection in File Menu

Click on the name of the simulation whose data you want to open.

### 5.1.11 Quit

*File* -> *Quit* will close GSimComposer.

## 5.2 Edit Menu

The **Edit** menu contains commands that pertain to editing activities in the *Editor* pane of the GSimComposer window during **Setup**.

These operations are most useful when editing the input file directly. See *Select Edit menu*.

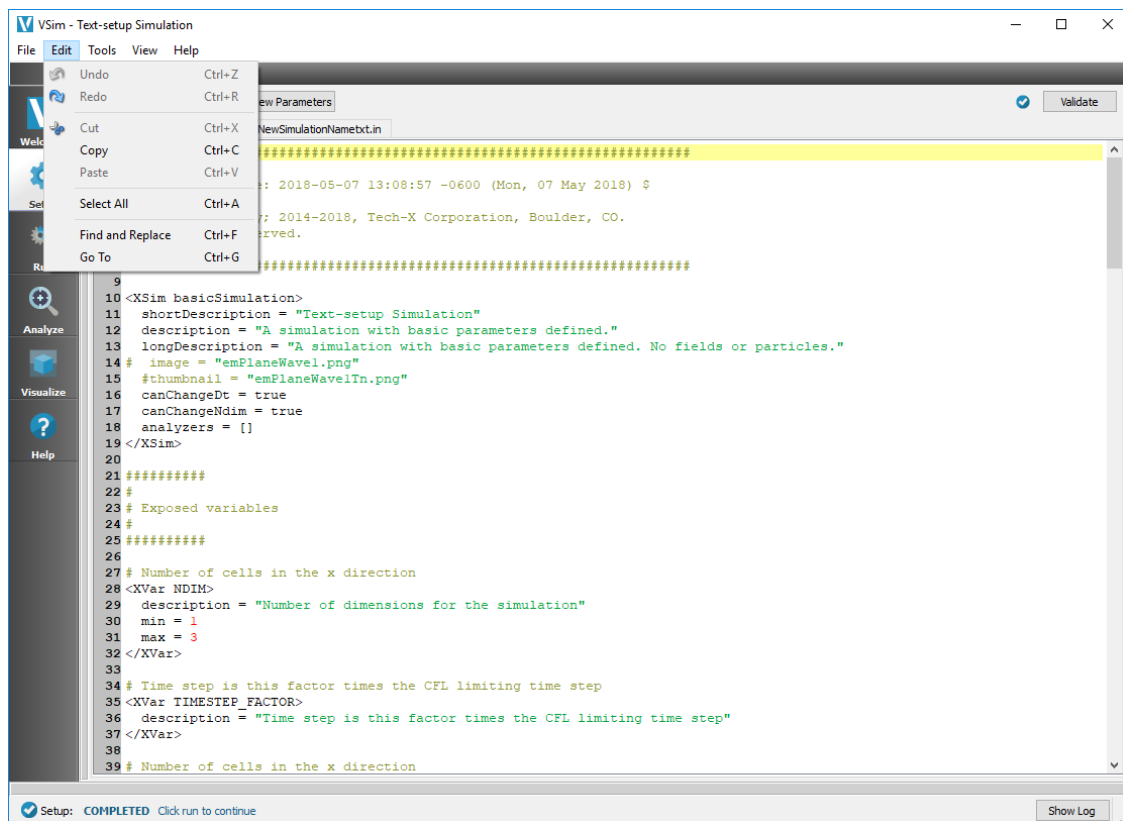


Fig. 5.5: Select Edit menu

## 5.3 View Menu

The **View** menu has the option for viewing the log file for GSimComposer. See [View Log](#).

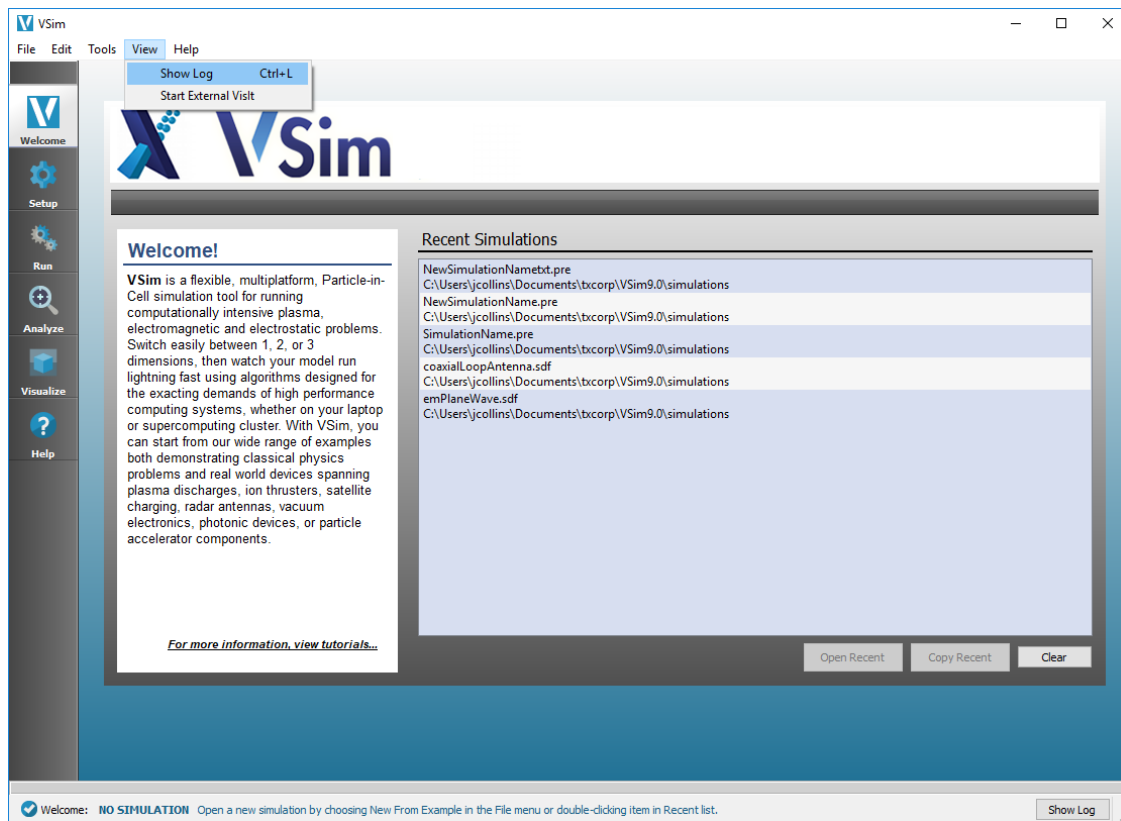


Fig. 5.6: View Log

This is useful if you run into problems and need help from the Tech-X Support team. It is a good idea to *Save to file* and send a copy of your log along with your message if you ever need assistance. See [Viewing the log contents from the view menu](#).

## 5.4 Help Menu

The **Help** menu contains both the VSIM Documentation labeled *Help Contents*. It also includes pop-up information *About GSim*. See [Accessing help through the help menu](#).

Selecting *Help Contents* is the same as selecting the **Help** icon in the left hand icon panel, and it opens the GSim Documentation Window in the users browser.

Selecting *About GSim* will bring up a pop-up menu that tells you information about the particular build of GSimComposer you have installed on your machine. It is very helpful to provide this information to Tech-X support personnel if you encounter any difficulties running GSim.

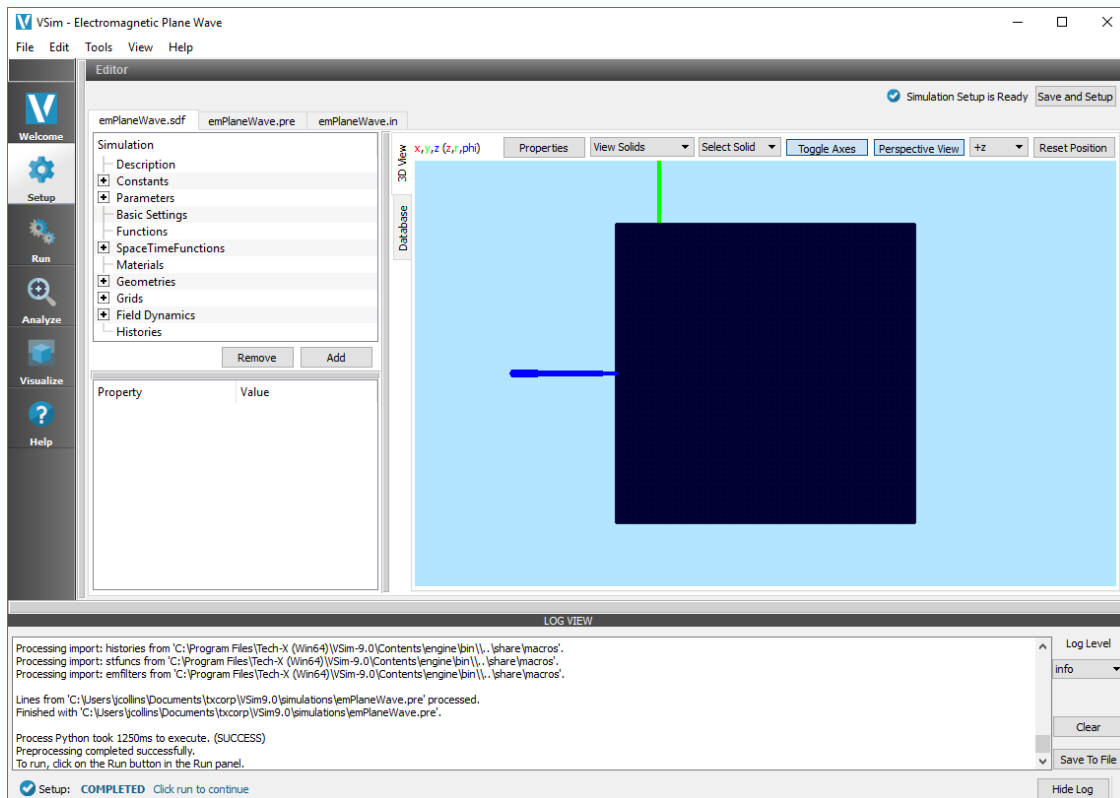


Fig. 5.7: Viewing the log contents from the view menu

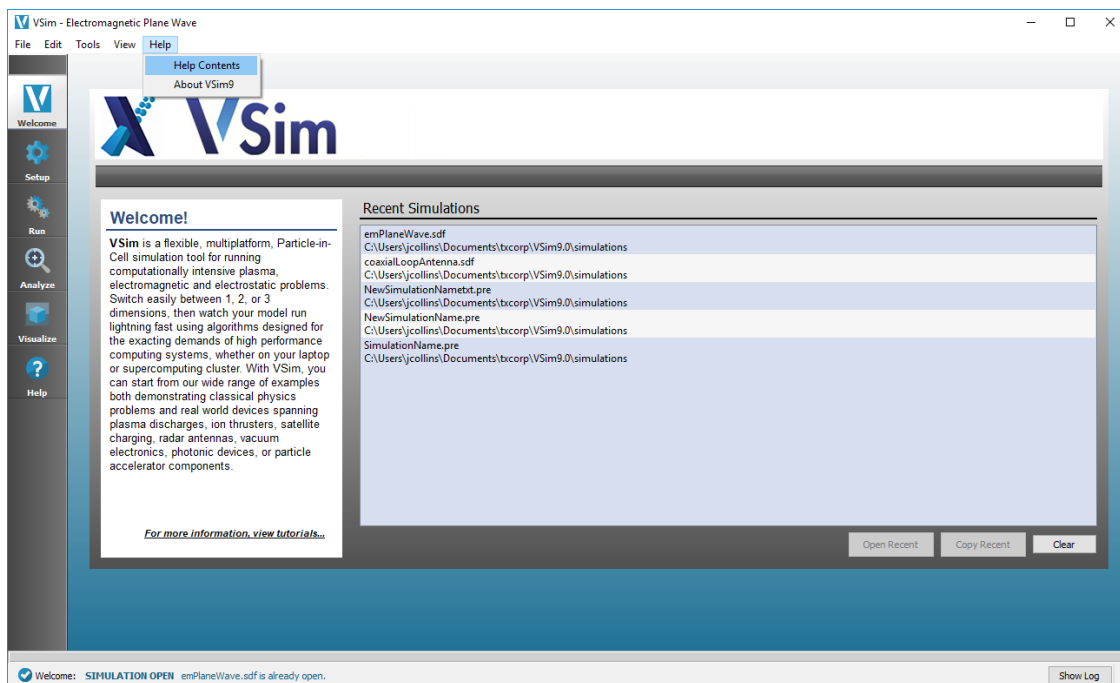


Fig. 5.8: Accessing help through the help menu

## 5.5 Composer Settings/Preferences Menu

### On Windows / Linux

The **Tools** menu provides access to global settings for Composer. The **Tools** menu contains the *Settings* selection.

Select *Settings* from the **Tools** menu to access the *Application Settings* window. See Fig. 5.9.

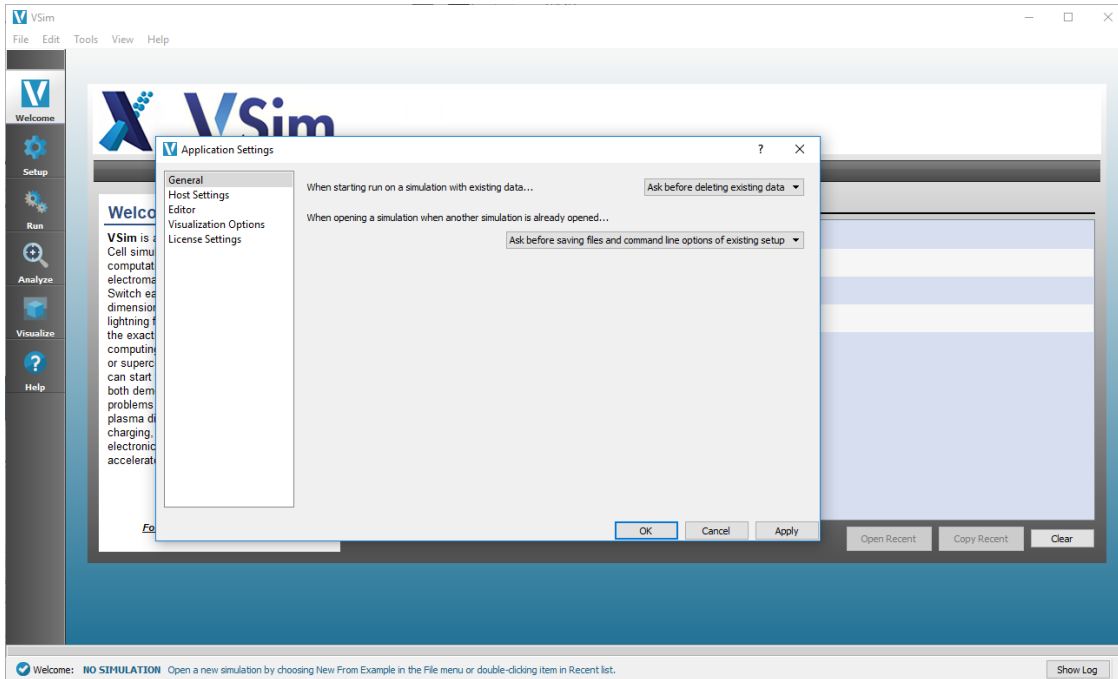


Fig. 5.9: Select Settings from the Tools menu

### On Mac

The *Composer* menu provides access to global settings for Composer. The *Composer* menu contains the *Preferences* selection.

Select *Preferences* from the **Composer** menu to access the *Application Settings* window. See Fig. 5.10.

### 5.5.1 General

General application settings set the default behavior for the Composer file, directory, and other actions. See Fig. 5.11.

- When starting run on a simulation with existing data, the default setting that Composer will use when starting a simulation that already contains data is *Ask before deleting existing data*. If you know that you will always want to create fresh data for each run, use the pulldown menu to set the default to *Always delete existing data*. If you know that you will always want to run on the data already available, use the pulldown menu to set the default to *Never delete existing data*. See Fig. 5.12.
- When opening a simulation when another simulation is already opened, the default setting that Composer will use is *Ask before saving files and command line options of existing setup*. If you know that you will always want to save the simulation, this can be switched to *Always save files and command line options of existing setup*. If you know that you will never want to save the simulation, this can be switched to *Never save files and command line options of existing setup*.

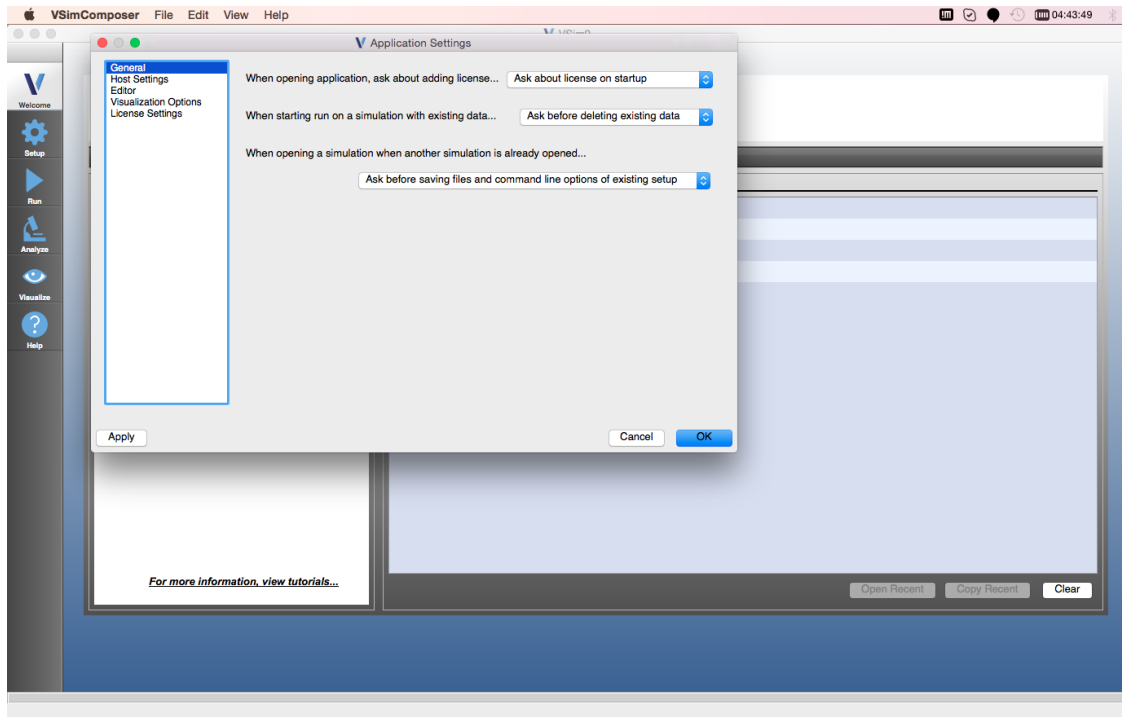


Fig. 5.10: Select Preferences from the Composer menu

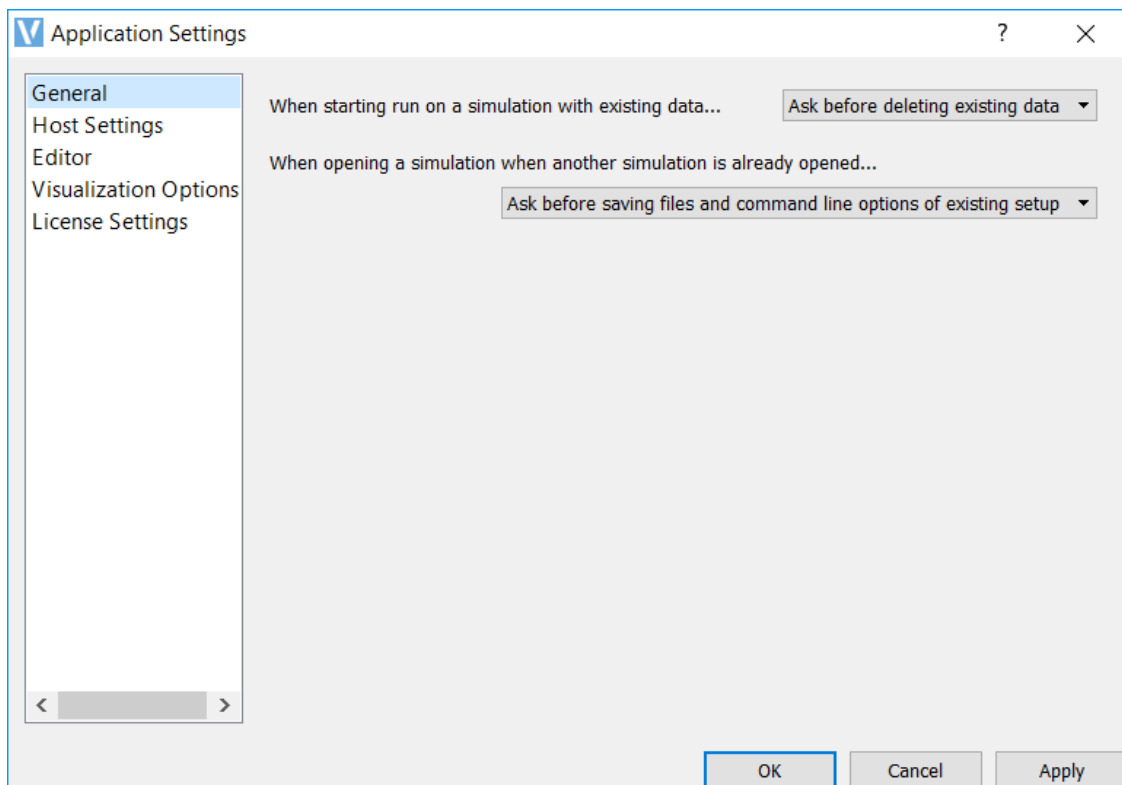


Fig. 5.11: General Application Settings



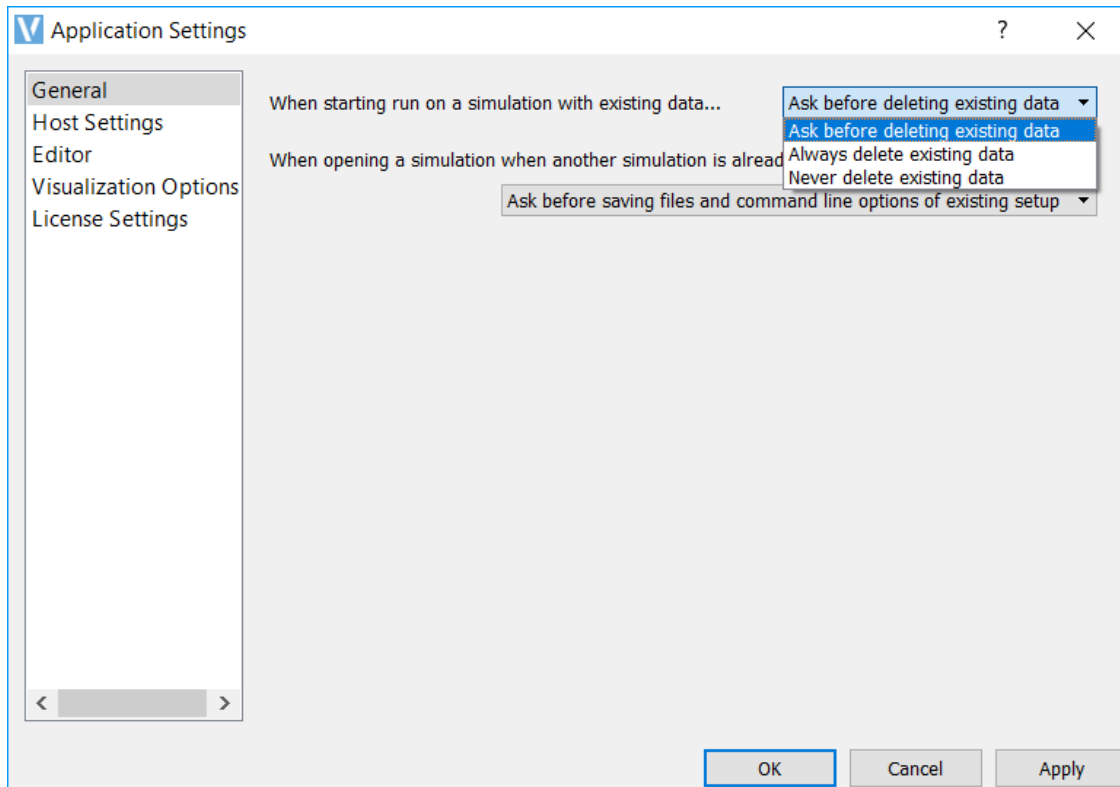


Fig. 5.12: Application Setting When Starting Run on a Simulation with Existing Data

### 5.5.2 Host Settings

The *Host Settings* section allows you to specify what machine to run on, the paths to your installation directory and workspace directory, and your preferences for serial or parallel simulations.

By default, you will be running on your localhost machine with the default installation directory and a preferred run method of serial.

#### General

Currently, GSim only allows for running simulations on the localhost. See Fig. 5.13.

#### Paths

- **Simulations directory**  
is the default directory for your runs.
- **Macros directory**  
houses the macros to be used in your runs.
- **Analyzers directory**  
houses the analyzers to be used in your runs.
- **Temporary directory**  
In some cases is necessary to set the environment variable TMPDIR (or TEMP on Windows) when execut-

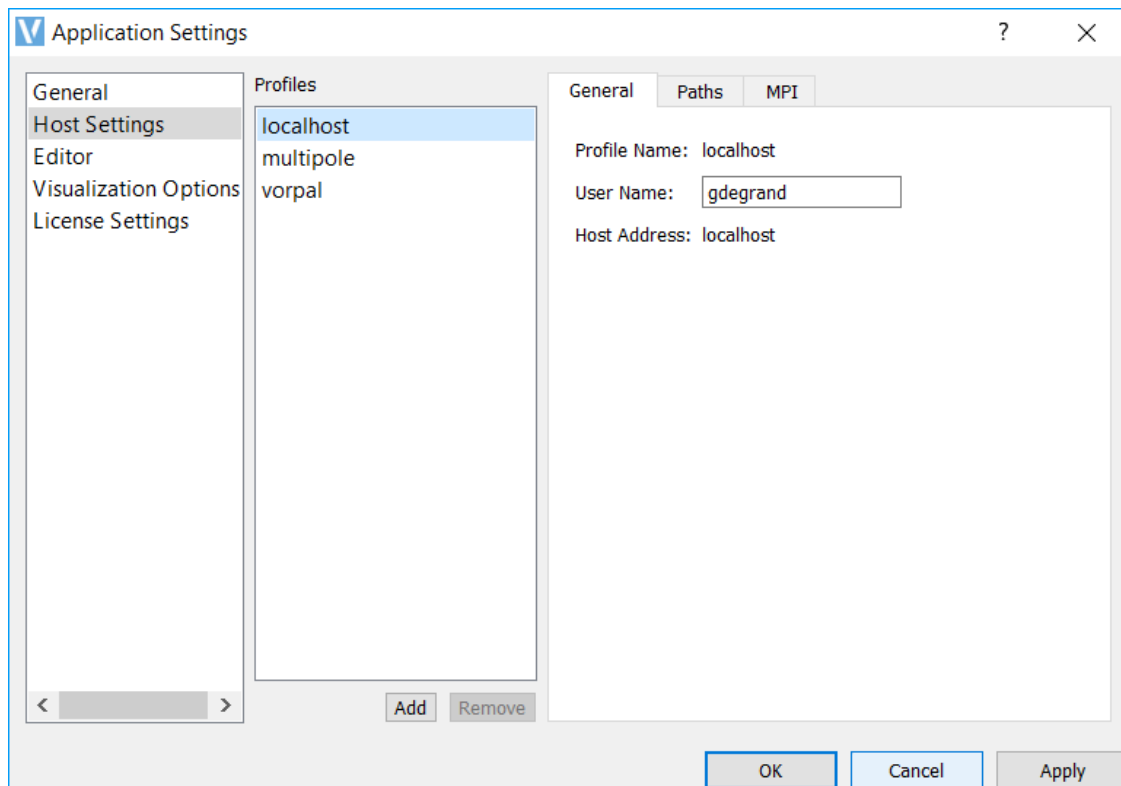


Fig. 5.13: Application Settings Host General Control Menu

ing vorpal. For example, on MAC when running in parallel and one gets an error “bind() failed on error Address already in use”, then setting the TMPDIR to a shorter string can solve the issue.

See Fig. 5.14.

## MPI

- Preferred run method is the GSim serial engine (vorpalsr) by default. If you have a multi-core system capable of parallel processing and a license activation file that is good on multiple cores, you can set the default to parallel instead of serial by clicking on the *Preferred Run Method* drop down menu and selecting parallel.
- Cores on machine shows the number of available cores for the current system that Composer detects.
- Preferred number of cores is the field in which you may enter a new value and change the number of cores that will run simulations. This is helpful when you would like to run simulations using fewer processors than the number of cores for which your software is licensed, or perhaps want to try load balancing using more processes than you have cores. When the value in the *Preferred Number of Cores* field is set to something other than the last saved value, Composer places an asterisk in front of the field label so that you are aware that you have changed the value and may wish to save the new value.
- Host File is where you can specify a file that contains the host nodes that you want to run on. This is useful if you have a large number of nodes, but need to run on a specific subset of them. For a description of how to create a hostfile see *Running Vorpalsr with mpiexec Using a Hostfile*.

See Fig. 5.15.

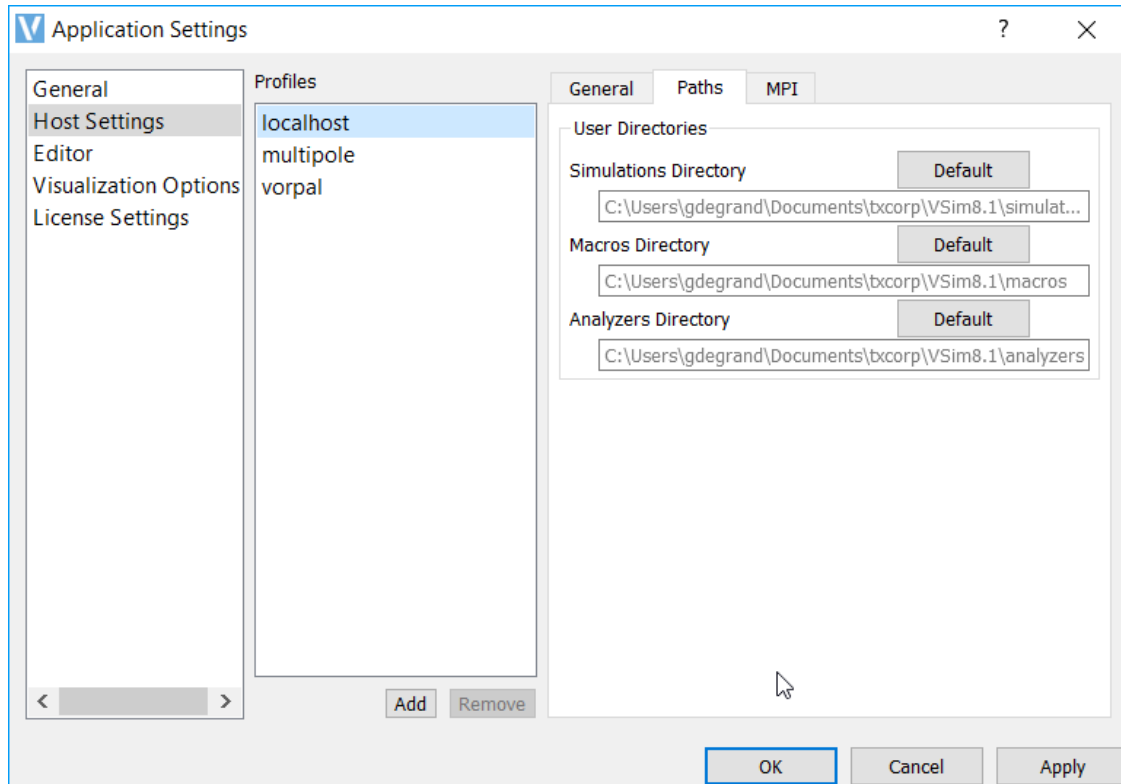


Fig. 5.14: Application Settings Host Paths Menu

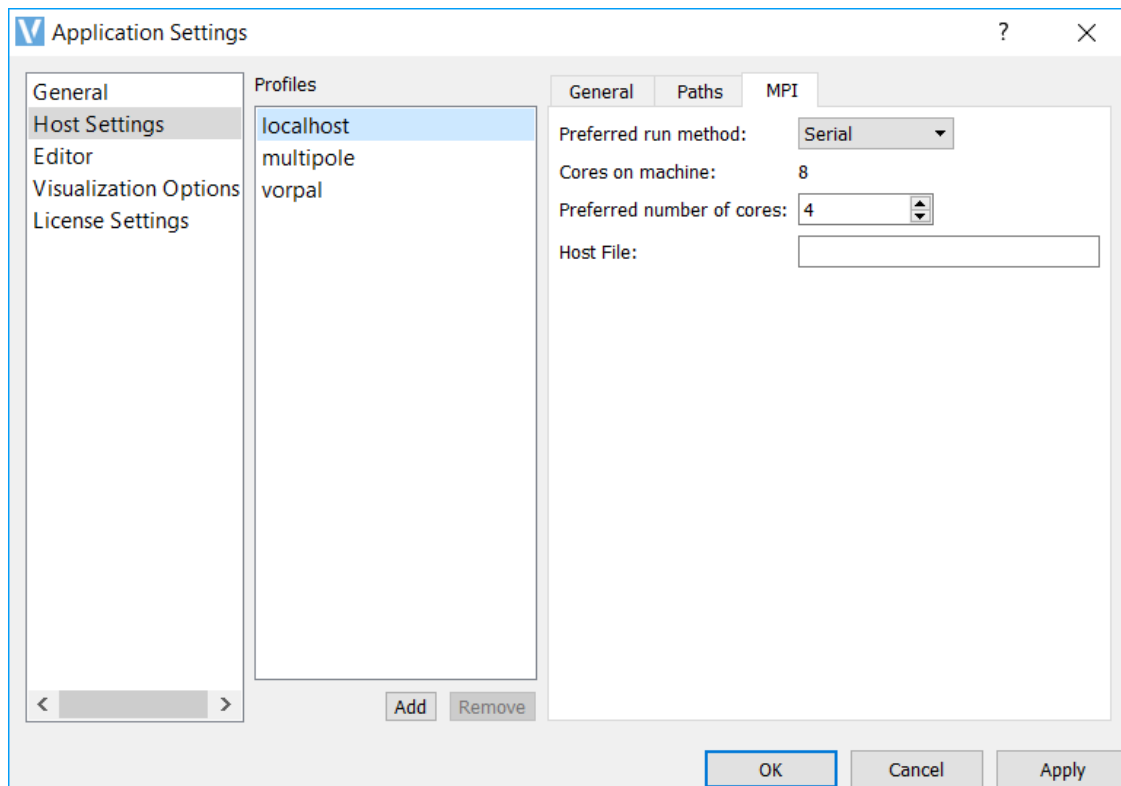


Fig. 5.15: Application Settings Host MPI Control Menu

### 5.5.3 Editor

The editor tab contains default settings for font size and a few other *Setup* tab options. These are editable to the users desired settings.

- **Files with Fixed-width Font**
  - Extensions refers to the file extensions that will obtain the following font and text size.
  - Font is where you can select the desired font.
  - Size is where you choose the desired text size.
- **All Other Files**
  - Font is where you select the desired font.
  - Size is where you choose the desired text size.
- Tabstop Width is the number of spaces that are inserted when the tab key is pressed.
- Color Style is where you select one of the choices for the color style of the text editor, including a white background, medium colored background, and dark background, with varying font colors.
- Open files in “Parameter” editor by default opens the file showing the editable parameters and an image overview (if this box is checked). When unchecked, the file is opened in the full text input file view.
- Use syntax highlighting, when checked, adds color to the text in the full input file view to help denote certain parts of the file. When unchecked, the text is all black.
- Show line numbers, if checked, shows the line numbers on the full text input file view. When unchecked, they are not.
- Highlight current line highlights the line where your cursor lies. If this box is unchecked, the line is not highlighted.
- Show post-processed file shows the post-processed .in file in a separate tab. This file shows the full text input file after it has gone through any Python calculations. When this option is unchecked, this file is not visible.
- Word wrap makes the text of the input file wrap at the end of the line. When this box is unchecked, the text will not be wrapped.

See Fig. 5.16.

### 5.5.4 Visualization Options

The visualization options tab allows the user control over default settings of the **Visualize** window in Composer.

- Manual font sizing allows you to control the size of the fonts of plots.
- Enable VisIt context menu enables you to right-click on a visualization and open VisIt itself, where the user can access every function and feature of VisIt. It also enables the embedded point and line tools in VisIt as well as some of the generic view controls.
- Try harder to load cycles and times determines how aggressive VisIt is when opening dataset. When this option is checked (ON), VisIt will open every single dataset looking for time and cycle information. When this option is unchecked (OFF), VisIt will only look at the first file in a series. The advantage of having this option OFF is that datasets with lots of files are opened more quickly and with less memory usage. The disadvantage of having this option OFF is that the dump slider will not display any time or cycle information— only the dump number.
- Default ColorTable is the default color table used for plotting color plots.

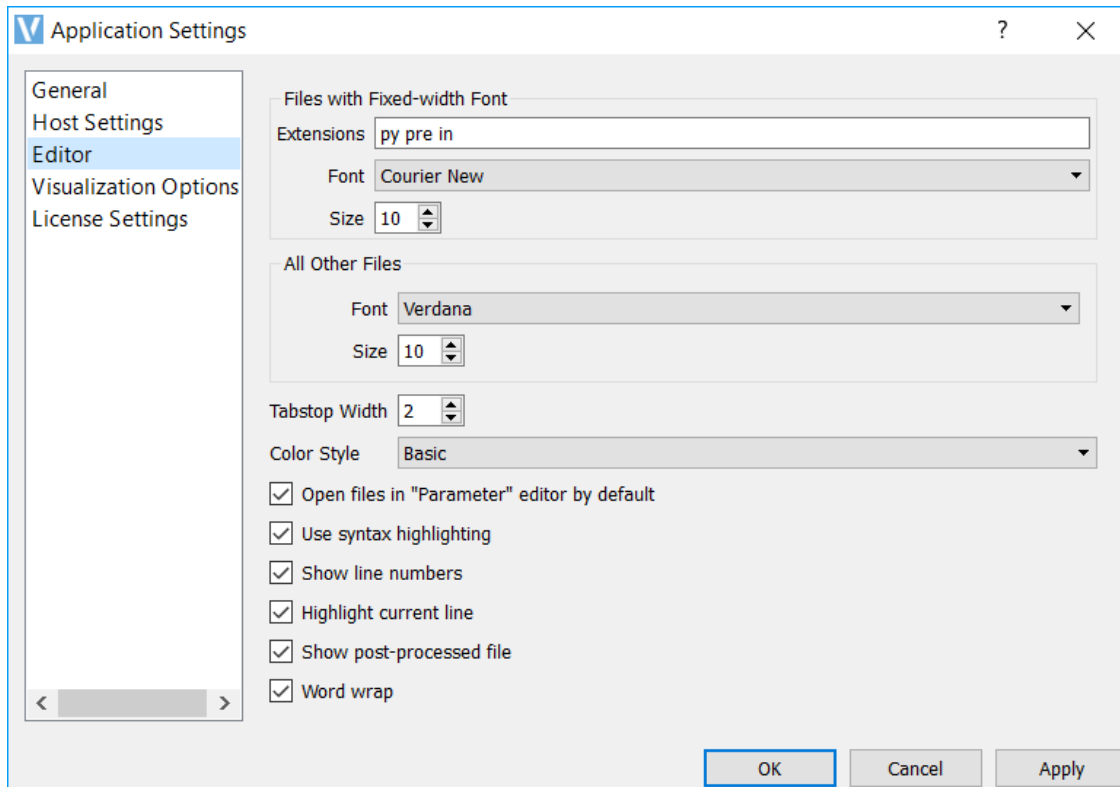


Fig. 5.16: Editor Menu

See Fig. 5.17.

For more information on VisIt, please see: <https://wci.llnl.gov/codes/visit/> and [http://www.visitusers.org/index.php?title=VisIt\\_Wiki](http://www.visitusers.org/index.php?title=VisIt_Wiki).

### 5.5.5 License Settings

See the Licensing section for help with licenses

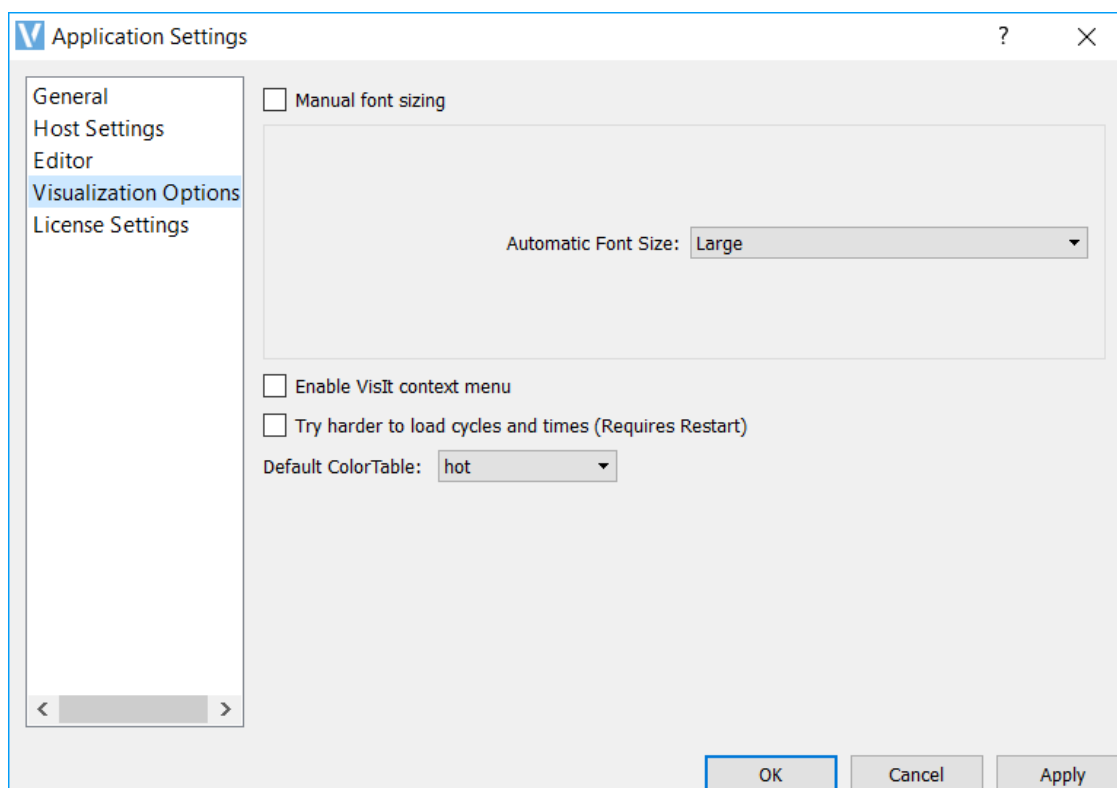


Fig. 5.17: Visualization Menu

## SIMULATION CONCEPTS

### 6.1 Simulation Concepts Introduction

GSim allows one to compute the dynamics of a system that has electromagnetic fields, and material shapes that are advanced dynamically a *time step* at a time. A GSim simulation can contain some or all of the objects, with them interacting in various ways.

The fields are defined on a structured grid in either cartesian or cylindrical coordinates. One can study just field dynamics, e.g., the propagation of electromagnetic fields on a grid, or solving for electrostatic fields for given boundary conditions and charge density, or fluid dynamics.

Material shapes, *geometries*, modify the dynamics of fields. For example, a conducting shape introduces an irregular region where the electric field vanishes. Consequently electromagnetic fields will scatter off of such a shape, and in electrostatics, such a shape will become an equipotential. Dielectrics shapes will modify the electric and magnetic.

To bring the power of many CPUs to simulation, GSim makes use of distributed memory (MPI) parallelism. In this method, the simulation region is divided into domains (domain decomposition), with each process containing both its domain plus some grid cells beyond. The additional grid cells are known as *guard cells*. They are used in the communication between processes. Additionally they are used as the locations of particle sinks. Particles that leave a simulation must be removed or reflected back into the simulation to prevent crashes caused by out-of-range accesses of memory.

Simulation results are analyzed by looking at the generated data. In the regular course of a simulation, the simulation data is periodically dumped. As well, GSim allows the definition of *Histories*, which are time sequences of data. Examples include the Poynting flux through a surface or the number of particles absorbed by a shape.

In this section, we will begin by going over simulation concepts and properties, including:

- Grids
  - Decomposition and guard cells
  - Periodic boundary conditions
- Geometries
- Fields
  - Electromagnetic fields
  - Electrostatic fields
  - Planar boundary conditions
  - Conformal boundaries
- Histories

## 6.2 Grids

The grids used by GSim are structured, coordinate aligned, where the grid lines are along coordinate directions. Such a two-dimensional grid is shown in *Structured 2D grid*. One can choose either a uniform spacing or a non-uniform spacing as shown in *Structured 2D grid*, and the coordinates may be either cartesian or cylindrical. In *Structured 2D grid*, each of the cells is numbered by its indices. In this 2D case, there are two indices; in general one for each direction. The cell indices start at 0 and end in the  $x$  direction at  $NX - 1$  for a grid that has  $NX$  cells in the  $x$  direction. For a 3D grid, there would be another direction out of the page.

0,NY-1					NX-1,NY-1
0,1	1,1				
0,0	1,0				NX-1,0

Fig. 6.1: Structured 2D grid.

A cell of a 3D grid is shown with  $z$  coming out of the page in two views in *3D cell in a view showing its edges and a view showing its faces*. A cell owns its interior plus the interior of its lower face in each direction plus, the interior of its lower edge in each direction, plus the lower node of its owned edges. The owned node is circled in both views of *3D cell in a view showing its edges and a view showing its faces*. The owned edges are shown on the left side of *3D cell in a view showing its edges and a view showing its faces*, with the  $x$ -edge red, the  $y$ -edge green, and the  $z$ -edge blue. Similarly, the owned faces of a cell are shown on the right side of *3D cell in a view showing its edges and a view showing its faces*, with the  $x$ -normal face red, the  $y$ -normal face green, and the  $z$ -normal face blue.

In FDTD EM, the concept of a dual grid is useful. The dual grid is the grid with nodes at the centers of the regular grid. The edges of the dual grid pierce the faces of the regular grid and vice-versa.



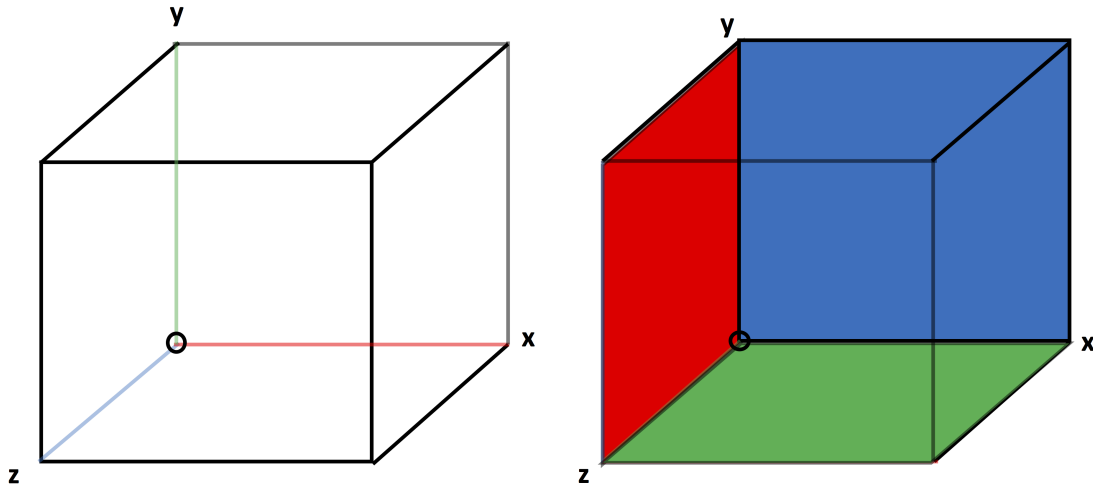


Fig. 6.2: 3D cell in a view showing its edges and a view showing its faces.

### 6.2.1 Guard cells

*Guard cells*, cells just outside the simulation grid, are needed for having sufficient field values in the simulation region, for particle boundary conditions, and for parallel communication (to be discussed later). An example of the first case is where a field must be known at each of the nodes of the simulation. Then, since the last node in any direction is owned by the cell one beyond the simulation, the cells one beyond the *physical grid* must be in the simulation. Thus, the grid must be extended by one cell in the last of each direction, as shown in *Grid extended to include the upper nodes, which belong to the cells one past the last cell in each direction.*

---

**Note:** The user-defined grid is called the physical domain. The grid extended by Vorpak is called the extended domain.

---

For particle boundary conditions, the grid must be further extended down by one cell in each direction. When a particle leaves the physical domain, it can end up in one of these additional cells, which can be either above or below. A data value associated with that cell determines what to do with the particle, e.g., absorb it (remove it), reflect it, or carry out some other process. The associated extended grid is shown in *Cartesian grid extended by Vorpak*. The physical cells are depicted by the red grid. The associated dimensions are in blue. The extended cells (shown in green) enclose both the physical cells and the guard cells added by Vorpak.

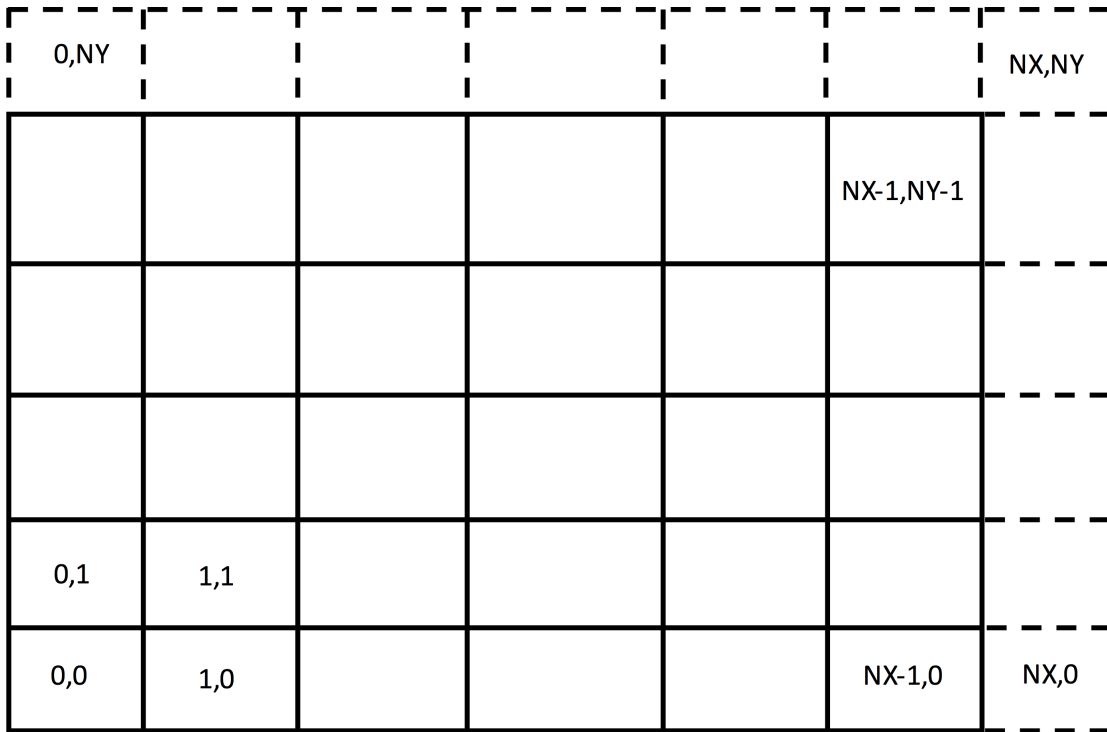


Fig. 6.3: Grid extended to include the upper nodes, which belong to the cells one past the last cell in each direction.

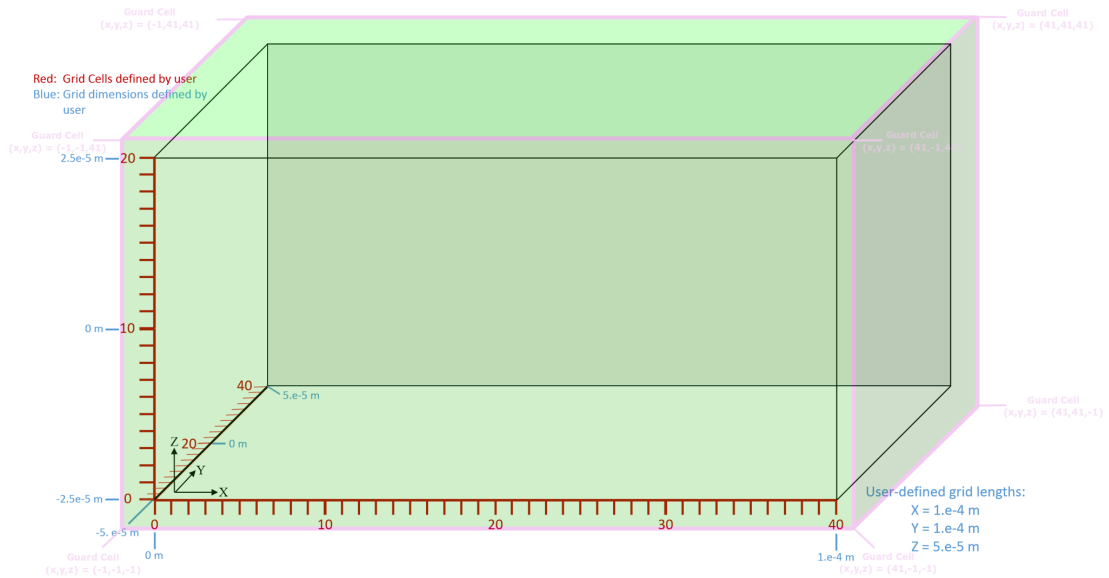


Fig. 6.4: Cartesian grid extended by Vorpil

## 6.2.2 Periodic Boundary Conditions

Periodic boundary conditions can be used to control both field and particle behavior at the edge of the simulation domain. In the case of particles, periodic boundaries ensure that particles leaving one side of the domain reappear at the opposite side. For example, particles traveling at a speed of  $-v_\phi$  will go through  $\phi = 0$  and reappear at  $\phi = 2\pi$ . Fields, on the other hand, will be copied from the plane at index 0 to the plane at  $NX$  and from the plane at  $NX - 1$  to the plane at  $-1$ , i.e. from the last physical cell to the guard cell on the other side.

## 6.2.3 Parallelism and Decomposition

Parallel (distributed memory, MPI) computation is carried out by domain decomposition. A particular decomposition is shown in *Parallel decomposition of a computational domain*. In this case, this is a decomposition of a rectangular region by the red lines, with the individual subdomains each give a unique index. However, Vorpai can simulate any region that is a non-overlapping collection of rectangles (appropriately generalized for 3D and 1D), with each rectangle being a subdomain of the decomposition.

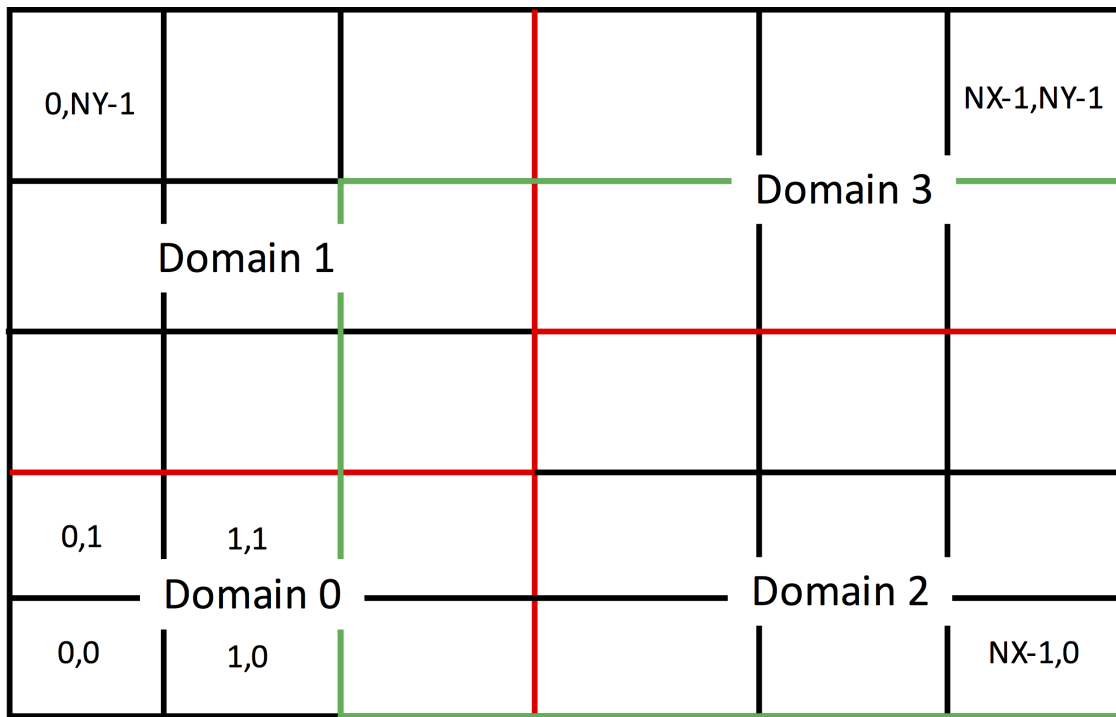


Fig. 6.5: Parallel decomposition of a computational domain.

For the most part, parallelism and decomposition are handled under the hood, but it is useful to understand a few concepts. In *Parallel decomposition of a computational domain*, one can see a green rectangle that extends one cell more into the simulation region beyond Domain 2. Fields in the cells of this overlap region are computed by the subdomain holding the cell, but they have to be communicated to Domain 2, as it needs this boundary region to update its fields on the next time step. On the other hand, particles may leave Domain 2 and end up in one of the cells still inside the green rectangle. Those particles must be sent to the processor holding the cell they are in for further computation.

For the above situation to work, the field update method for a given cell must not need information more than one cell away. The standard updates for electromagnetics and fluids indeed have this property. As well, the particles must not travel more than one cell in a time step. This is true for explicit electromagnetic PIC with relativistic particles, as the Courant condition prevents the time step from being larger than the time it takes for light to cross any cell dimension, and relativistic particles travel slower than the speed of light. Finally, the particles must not interpolate from fields more

than one cell away, nor must they deposit current more than one cell away. This is true for the simplest interpolation and deposition methods.

However, if you have electrostatic particles that travel more than one cell per timestep, or particles that have a larger deposition footprint, then manual setting of some parameters may be necessary. In particular, the grid parameter, `maxCellXings`, states the maximum number of cells a particle might cross in a simulation, and the parameter, `maxIntDepHalfwidth`, provides the width of the deposition stencil. From these follow the overlap needed for the subdomains. These parameters are discussed in more detail later.

## 6.3 Geometries

Geometries in Vorpal are non-grid-aligned material shapes. They can be defined in a number of ways, e.g., a triangular surface mesh from an STL file, a set of shape from a STEP file, Constructive Solid Geometry (CSG), and functional (a function defining whether a point in space is inside or outside). In visual setup, the material assigned to the shape determines how the electromagnetic field interacts with the shape.

Visual Setup supports STL import with translation, STEP import, and CSG. One can then assign a material to the shape. CSG is discussed in the *Visual Setup* section, where you learn to build primitives and perform operations on them.

In text setup, you can still use CSG primitives, but the process is a little different. Rather than being able to simply click and add primitives, text-setup requires one to import the appropriate `geometries` macro and define primitives using built-in functions. This all will be covered in detail in the “Geometries” section in the Reference Manual. Also covered in this section are the procedures for importing CAD and Python-defined geometries, moving and rotating shapes, and building custom primitives for your simulation.

One can also use geometries in particle boundary conditions. Particles can be emitted, reflected, or absorbed by a geometry. Additionally, an absorber on a geometry can be attached to a secondary emitter or a sputterer as a source for the same or another kind of particle.

## 6.4 Electric and Magnetic Fields

The electric and magnetic fields are the most important fields in GSim, as they impact the motion of charged particles. In this section we discuss how fields generally work, then we discuss the field update concept. We then discuss the particulars of the updates for each of electromagnetics and electrostatics.

### 6.4.1 Field Basics

The general concept of a field is a scalar, vector, or tensor that is a function of space and time. It is most commonly implemented in GSim by values on a grid, i.e., a value for each cell. For finite difference methods, the different components (e.g., vector components) have a location, i.e., place where they most accurately represent the field value, within each cell. For a scalar field, like the electrostatic potential, the location is either at the node (lower corner) or the cell center. For a vector field, the natural locations are either at the centers of the edges or at the centers of the faces. Hence, a field in GSim has a property, `offset`, that determines this offset.

Fields can be messaged as needed and described in *Parallelism and Decomposition*. In GSim, fields are messaged according to their overlap, and they are always messaged both up and down.

Some fields are used for deposition of charge and/or current. These deposition fields come with a few changes. First, they are automatically zeroed at the beginning of each time step. Second, to get the charge and current correct in a parallel simulation, the contributions in the guard cells on one domain have to be sent to the owning domain and be added into the charge or current on that domain.

### 6.4.2 Field Updating Basics

Fields can be either static or dynamic. E.g., the magnetic field in an electrostatic simulation does not evolve. It is set once, and that field is used throughout the simulation. On the other hand, the electric field in an electromagnetic or electrostatic simulation changes at each time step, and the magnetic field in an electromagnetic simulation also changes at each time step. In addition, a field may be made up of a static part and a dynamic part, in which case the two are added together to get the total field at each time step.

The update of a field can be either explicit or implicit. Explicit means that one can write the new field at a given cell in terms of the old field values. Implicit means that one must solve an equation for the new field values. An example of the latter, to be discussed in more detail. In either case, one is updating the field, and the object that does this is known as an *updater*.

### 6.4.3 Electromagnetics

In electromagnetic simulations the electric and magnetic fields obey Maxwell's equations, i.e., Ampere,

$$\frac{d\vec{E}}{dt} = -\frac{1}{\epsilon_0}\vec{J} + \frac{1}{c^2}\nabla \times \vec{B}$$

and Faraday.

$$\frac{d\vec{B}}{dt} = -\nabla \times \vec{E}.$$

These are discretized and updated using the Yee algorithm [Yee66]. In that algorithm staggered grids are used, which is equivalent to saying that the electric field is an edge field, and the magnetic field is a face field, following *Grids*. The Yee algorithm can be thought of as using the integral formulation of Maxwell's equations on the faces of the grid (for  $B$ ) and the faces of the dual grid (for  $E$ ). The corresponding updaters are the *YeeAmpere* and the *YeeFaraday* updaters. The update for pure EM is staggered in time, as in leap-frog integration.

The update region is over the interior of the simulation. This varies for different components of the electric field and is illustrated in *Update region for the electric field*. The electric field in the  $x$  direction, shown in red, lies on  $x$  edges. The interior  $x$  edges have lower-left cell of  $(0,1)$  and upper-right cell of  $(NX-1,NY-1)$ . In Vorpai, the region is greater than or equal to the lower bounds and less than (not equal to) the upper bounds. Hence, the update slab for  $E_x$  is  $[(0,1),(NX,NY)]$ . By similar reasoning, now referring to the green arrows, the update slab for  $E_y$  is  $[(1,0),(NX,NY)]$ .

For the magnetic fields, there is a similar difference in the update region per component. The magnetic fields are face fields, and so they are updated on any face that is in the interior or on a boundary.

When dielectrics are present, Maxwell's equations need to be modified. Ampere's equation gives the new value of the displacement,  $D$ , from which one obtains the new value of the electric field,  $E$  by multiplication by an effective inverse dielectric tensor. Accurate algorithms for the time domain are described in [WBC13, WC07], while a more rigorous, but frequency domain algorithm is described in [BWC11].

### Electromagnetic Slab Boundary Conditions

Beyond the edges of the simulation, values of the electric field are not known. Hence, one cannot update the electric field at the edge, as that would require a difference with an unknown value. Instead one sets boundary conditions. For the simple case of a purely rectangular region, the boundaries that must be set are shown in *Slab boundary regions for the electric field*.

They are simply the tangential values of the electric field on the boundary. Just as in continuum EM, one need not set boundary conditions on the magnetic field. Setting the values of the tangential electric field at boundaries is sufficient.

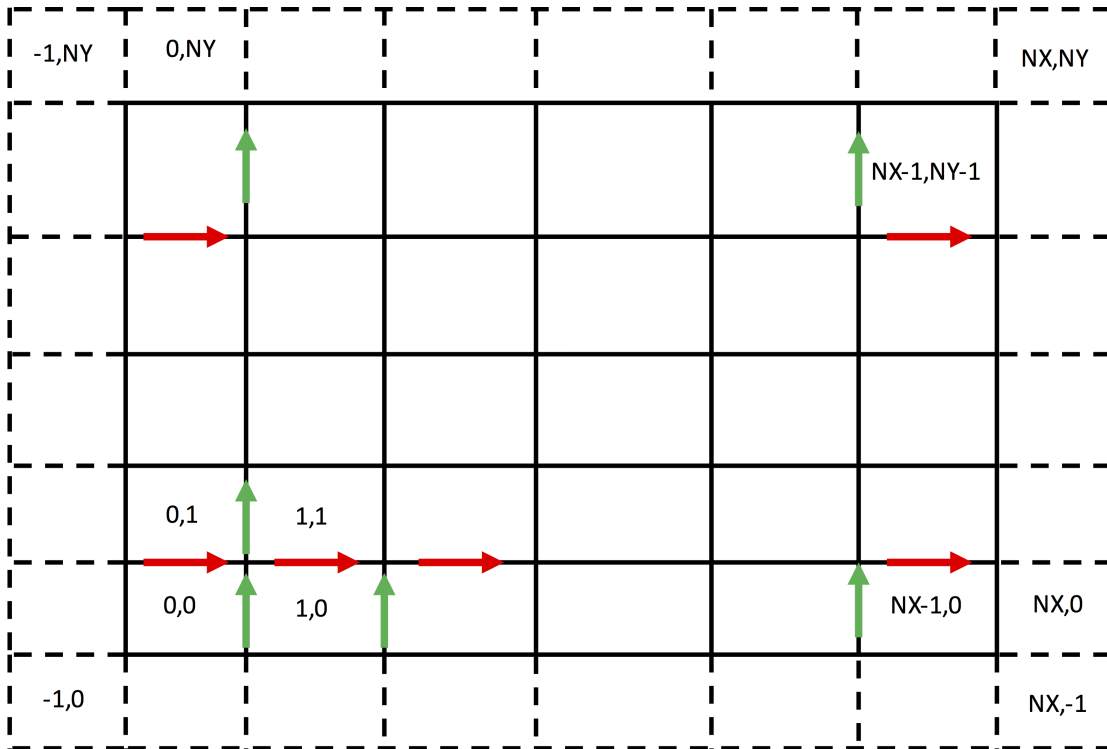


Fig. 6.6: Update region for the electric field.

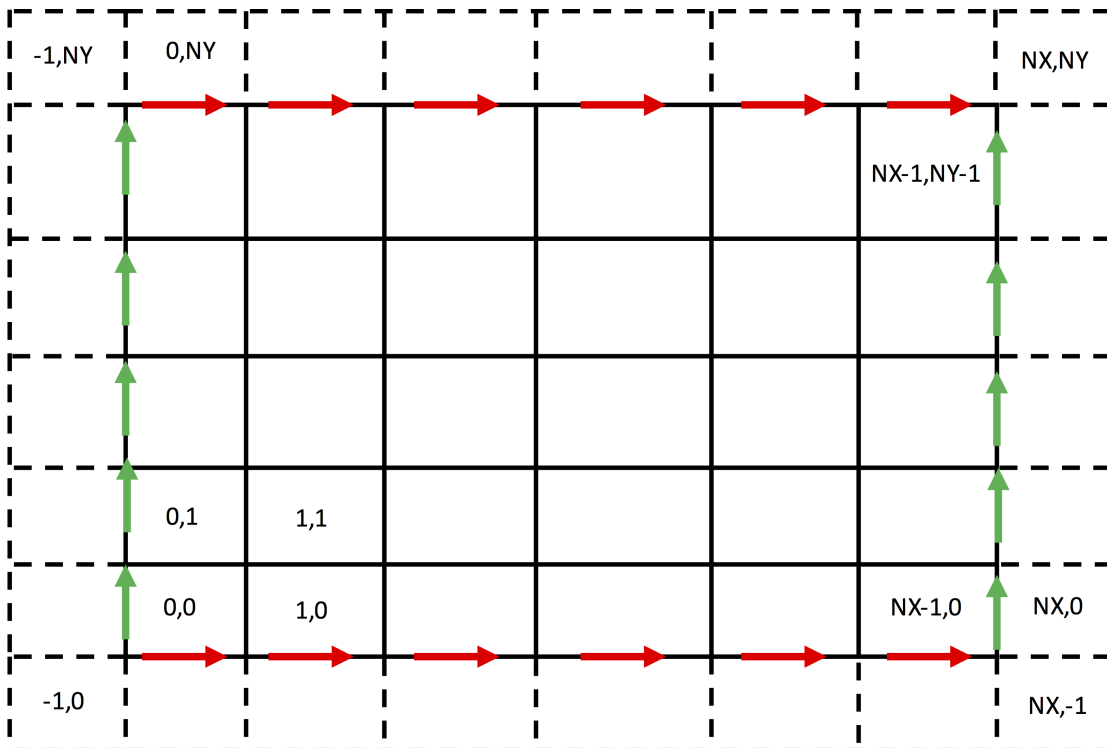


Fig. 6.7: Slab boundary regions for the electric field.

## Electromagnetic Conformal Boundary Conditions

Electromagnetic boundary conditions are to some degree setting the value of the electric field, but they can also involve a modification of how the magnetic field is updated.

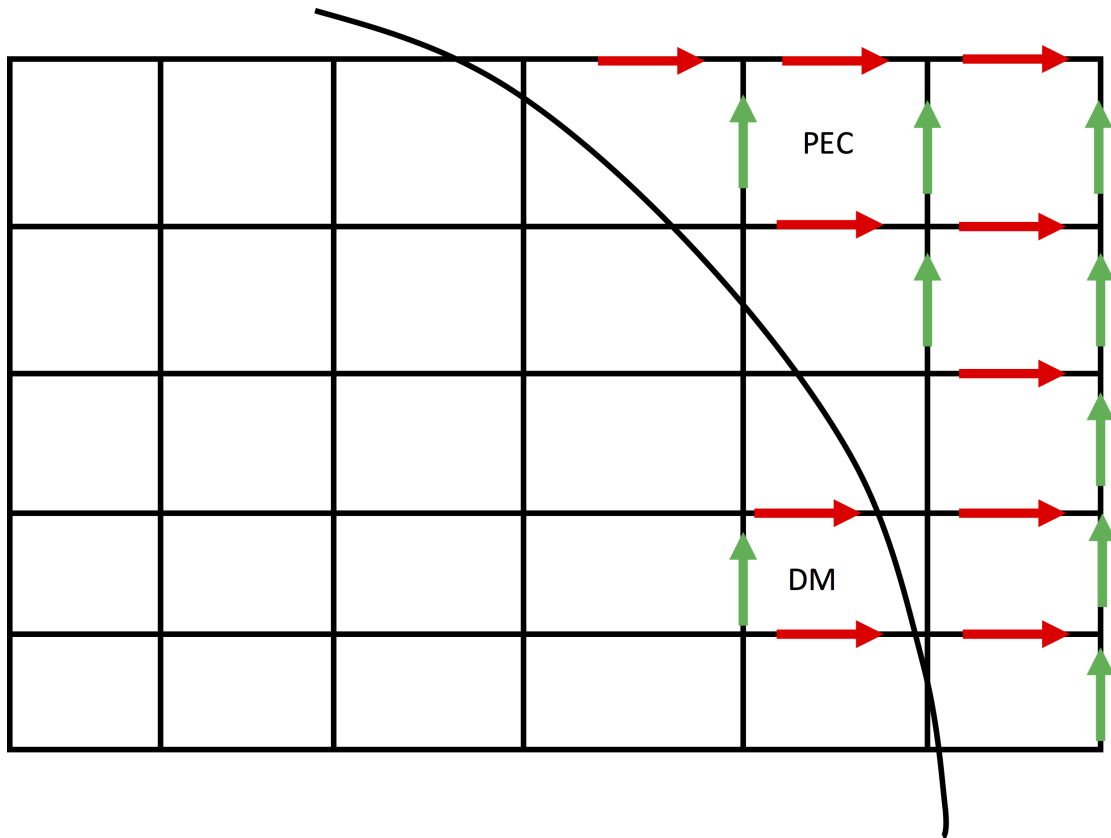


Fig. 6.8: Conformal boundaries for the electric field.

In *Conformal boundaries for the electric field*, is shown a curve the interior (lower and left) of which is vacuum, while outside is Perfect Electric Conductor (PEC). All electric fields on edges that are totally in the PEC are set to zero. Then there are two approximations. In the *stair-step* approximation, one additionally sets to zero all electric fields whose edges are more than half outside. In the *Dey-Mitra* [DM97] algorithm, one starts by keeping any electric field on an edge even partially outside of the PEC. One then updates the magnetic field by using a line integral around the cell to get the change in magnetic flux, and then dividing that by the area of the part of the cell outside of the PEC. For the cell marked DM in *Conformal boundaries for the electric field*, this would consist of adding up the marked electric fields (two in the x-direction and one in the y-direction) with appropriate signs. Because of the area divisor, the Dey-Mitra algorithm can be unstable for time steps smaller than the uniform-grid CFL time step limit, and the smaller the area of the cell, the more the time step is limited. In practice, one sets an amount of acceptable reduction of the time step, and then one can compute the fractional cell areas that must be dropped [NCW+09].

### 6.4.4 Electrostatics

Electrostatics refers to finding the fields by a different mechanism. The electric and magnetic fields are still present. However, the magnetic field is static and imported, while the electric field at each time step is found by first solving for the potential, which satisfies Poisson's equation,

$$-\nabla \cdot \epsilon(\nabla \phi) = \rho,$$

and then finding the electric field from

$$\vec{E} = -\nabla \phi,$$

which becomes finite differencing in numerics. Because we cannot directly state the solution for the potential,  $\phi$ , but instead we have to solve for the potential, this is an implicit update. Therefore, solving Poisson's equation involves setting up finite difference equations which leads to a system of equations. The system of equations is solved by inverting a matrix at each time step and solving the unknown ( $\phi$ ) using the source term ( $\rho$ ). The default value of the dielectric constant ( $\epsilon$ ) corresponds to that of a vacuum. However, *SpaceTimeFunctions* can be defined to specify regions in which the dielectric constant differs from a vacuum, which can then be inserted into the Poisson Solve.

#### Electrostatic Slab Boundary Conditions

Poisson's equation, upon discretization connects  $2*D+1$  grid nodes, as shown by the nodes within the curve in [Conformal boundaries for the electric field](#). Because this *stencil* reaches to each side of the node, it cannot be applied to edge nodes (covered by open circles). On those grids one must apply boundary conditions. For Dirichlet boundary conditions, one specifies the value of the potential on that node.

For Neumann boundary conditions, one specifies the value for the difference between the value of the potential on a node and the value on the node just interior. In general one must take care not to specify a mathematically impossible situation. As applied to Neumann boundary conditions, one cannot, e.g., specify zero Neumann boundary conditions on all surfaces while having non-zero net charge in the interior, as that would violate Gauss's law. Similarly, if one has a simulation that is periodic in all directions, consistency requires that it contain no net charge. Further, the matrix is singular unless one sets the value of the potential on at least one node.

#### Electrostatic Conformal Boundary Conditions

As this is an extensive subject, we simply say that for the nodes interior to a surface of constant potential, one specifies that the potential on that node, rather than being related to nearby points, is simply given. Otherwise one constructs the matrix as before.

#### Solving Poisson's equation

Upon discretization and applying all boundary conditions, numerically one is left with a large matrix equation to be solved. There are multiple ways to do this within GSim, with direct or iterative solvers. This is discussed in more detail in [Selecting Solvers and Solver Parameters](#).



## 6.5 Particles

Particles can be represented by macroparticles or a fluid. Macroparticles should be used when there is a need to capture kinetic effects. They also have more features in VSim.

### 6.5.1 Macroparticles

Macroparticles allow one to model kinetics (velocity distributions) for physical particles. Macroparticles are composed of a certain number of physical particles. The number of physical particles to represent in a single macroparticle is determined by factors such as the physical particle density, volume of a grid cell, and the number of macroparticles in a simulation grid cell. Macroparticles, if used so that there is accurate resolution, produce the same result as would simulating all individual physical particles, but with significantly higher computational speed. We will elaborate more on macroparticle definition and effects on simulation resolution in later sections. For macroparticles there are a number of options for particle loading and emitting from sources, as well as various types of particle sinks available.

There exist over 70 variations in particle type and evaluation, including:

- Boris (Relativistic, Non-relativistic, Tagged, Weighted)
- Electrostatic
- Cylindrical species

For more information on particle species and some of these other algorithms, please visit the “Species” section in the Reference manual. More information on variable and managed weights can be found in [Particle Weights](#).

### Species and their kinds

A common term in VSim for macroparticle is Species, as that is the type of block that defines a set of macroparticles. VSim allows you to specify a species kind, determined by the particle type (relativistic, electrostatic, etc.). Within these kinds are options for particles (i.e hydrogen, helium, argon, xenon, etc.) whose charge, mass, cross-section, and other relevant data sets are built into VSim. Alternatively, you can import your own species data in VSim for more custom simulations, so long as the necessary particle information is successfully imported to VSim.

Please visit the “Species Kinds” section in Reference manual for all kind options and further details, as well as information on importing your own particle data.

### Particle Sources

VSim allows for the implementation of both primary and secondary particle sources, where “primary” refers to initial particles that are introduced into the system, and “secondary” to particles that are created by interactions between these primary particles and either other particles or metal surfaces in the simulation.

Particle “loading” refers to the placement of particles volumetrically in a vacuum, whereas particle “emitting” is the ejection of particles from the surface of a metal.

One can load particles over time in one of the following ways:

- Load all at once. This method is required in electromagnetics simulations for charge conservation.
- Load over about ~1000 time steps
- Load in some custom way using your own external files (.txt files in hdf5 format)

Particle sources in 2D ZR cylindrical coordinates require extra precautions, as the loading algorithm will want to distribute particles uniformly in both directions without accounting for the larger volume/area at large radii that occurs

in this coordinate system. To learn how to compensate for this phenomenon, please visit the “Working in Cylindrical Coordinates” section in the Reference manual.

## Particle Sinks

Particle sinks, on the other hand, generally remove particles from a simulation or from a region therein. There are two basic types of sinks, with many more types of physical sinks available for use in your simulations.

- **Messaging sinks**  
Automatically established by Vorpai, they are used to communicate particles between processors in parallel runs, enforce periodic boundary conditions, etc.
- **Physical sinks**  
Physical sinks can remove particles from a region of the simulation, absorb incident particles on a boundary, or even perform more specialized tasks.

Particle sinks typically involve at least upper and lower bounds, and at the most basic level must enclose the entirety of the simulation space that contains particles. If this is not the case and your particles try to travel into regions not included in the simulation grid, Vorpai will likely crash. In the case of periodic boundary conditions, particles “wrap around” from one side of the simulation to the other.

## 6.5.2 Fluids

Particles can also be represented by fluids. The fluid representation is valid in the limits when the pressure is known. The two cases where this is valid are cold fluids, where the pressure vanishes, and at high collisionality, so that the pressure can be obtained from an equation of state (EOS), such as the adiabatic EOS. The available fluid kinds

- Cold fluid
- Euler fluid
- Neutral Gas

Only the most basic fluid dynamics is supported. There are no conformal boundary conditions nor any internal boundary conditions of any kinds.

## 6.6 Reactions

Reactions are bulk processes that occur on time scales much shorter than the time step of the simulation. As an example, collisions between atoms or electrons and atoms occur on times of the atomic unit time,  $2.4 \times 10^{-17}$  s, while even in laser-plasma interactions, the laser period is typically of order  $3 \times 10^{-15}$  s. Hence, even within a time step with the shortest time scales modeled by PIC methods, a collision can be considered an instantaneous process. This is better for plasma discharges that evolve on ms time scales or microwave devices for which the time scale is ns.

The reactions that VSim contains are

- Particle-Particle Collisions
- Particle-Fluid Collisions
- Three-Body Reactions
- Field-Ionization Processes
- Decay Processes

### 6.6.1 Reactions Implementation

Decay processes are relatively easy, as they involve only a single particle at a time. For particle-particle collisions, VSim uses Direct Simulation Monte Carlo methods, in which one computes the collisions between the pairs within each cell. Three-body reactions are primarily useful for recombination. Finally, field ionization is the creation of an electron-ion pair from a neutral atom in a strong electric field.

Direct Simulation Monte Carlo takes into account that random interactions between particles occur with non-negligible probability only when the particles are in close proximity. To avoid checking the  $N^2$  distances between  $N$  interacting macroparticles, the VSim MonteCarloInteractions package limits interactions to those between macroparticles within the same cell. This reduces the number of possible interactions to

$$N_{\text{cells}}(N_{\text{ppc}})^2$$

where  $N_{\text{cells}}$  is the number of cells in the simulation and  $N_{\text{ppc}}$  is the number of particles per cell. Hence,

$$N = N_{\text{ppc}}N_{\text{cells}}$$

and

$$N^2 = (N_{\text{cells}})^2(N_{\text{ppc}})^2 \gg N_{\text{cells}}(N_{\text{ppc}})^2$$

VSim has three reaction frameworks. The original *reduced* reaction framework had only a limited number of reactions. The *monte carlo* framework has a larger set of reactions. The *reactions* framework is now preferred, as it has an even larger set of reactions as well as using the No-Time-Counter method for algorithmic speedup.

### 6.6.2 Resolution Issues with Reactions

In order to accurately model the desired physics, one must resolve the physical distributions of the interacting particles and the temporal evolution of the distributions. For sufficient spatial resolution the number of macroparticles in the simulation must be large enough to smoothly resolve the spatial distributions of the species.

As for temporal resolution, each Monte Carlo interaction has an intrinsic time scale set by the physics of the interaction itself. In other words, the probability for an interaction event to occur can be written as  $P = dt/T_i$ , where  $dt$  is the simulation time step and  $T_i$  is the natural time scale associated with the interaction itself.

The fundamental probability for an interaction event to occur between two macroparticles in a given cell with volume  $V$  in a time step  $dt$  is

$$P = \frac{N_1 N_2 \sigma(v) v dt}{N_x V}$$

where  $v$  is the relative velocity between the two macroparticles,  $N_1$  and  $N_2$  are the numbers of physical particles per macroparticle in the final-state species. This implies that the natural time scale associated with this numerical process is:

$$T_i = \frac{N_x V}{N_1 N_2 \sigma(v) v}$$

If the simulation time step  $dt$  is not small enough to resolve the natural interaction time scale, then inaccurate statistics will result.

The last issue of resolution comes from the need to accurately sample the velocity distributions of the interacting particles. Since the probability for an interaction event to occur non-trivially depends on the particle velocities, and since a single macroparticle samples only one point in velocity space, accurate statistics may only be achieved in some simulations with many macroparticles per cell, such that the velocity distributions of the participating species are well sampled within each cell.

## 6.7 Macroparticle Weights

As discussed in *macroparticles*, simulation particles in VSim can represent any number of physical particles. This ratio of physical to macro particles is called the weight of the particle, and in VSim simulations can be run with either *constant* or *variable* weights. The benefit of constant weight particles is that they are simpler, reactions with them do not cause an increase in the number of macroparticles during the simulation, and one can easily relate the physical density to macroparticle density.

However, there are drawbacks to constant density simulations, namely when the simulation also contains reactions. As discussed in *Reactions*, statistics are important for Monte Carlo-based reactions (all reactions within VSim), and so maintaining a minimum number of particles per cell is required for reaction accuracy (in general this number is 10 or more). So an ideal simulation would have homogenous macroparticle density regardless of the underlying physical density. This makes variable weights preferable over constant weight simulations when:

- Physical density is irregular, such that constant weight particle would need to vary in number more than an order of magnitude from one location to another within the simulation domain
- Using cylindrical coordinates, as the equilibrium macroparticle distribution is heterogeneous.

This is not to say that variable weight simulations are perfect, just preferred. Variable weight simulations have the following drawbacks:

- Particles with very small weights can have little effect on the local field values, while taking just as much computational resources as large weighted particles
- Particles with very large weight particles can disrupt local field values
- Reactions with variable weight particles will split the largest particle into two pieces, automatically increasing the number of macroparticles in the simulation even for simple reactions like elastic scattering

These issues can be solved via Splitters and Combiners using *Managed Weights*.

### 6.7.1 Managed Weights

An ideal PIC simulation will:

- Have homogenous macroparticle density
- Have a constant (in time) total number of macroparticles
- Have enough particles-per-cell to maintain good reaction statistics

Neither constant nor variable weight particles can satisfy all these conditions in any non-trivial physics simulation. This is where *Managed Weights* can significantly improve both performance and accuracy of the simulation.

*Managed Weights* uses algorithms that split and combine macroparticles to attempt to keep the particles-per-cell constant in each cell, while also preventing the weights from becoming too large or too small. Splitters, as their name suggests, take a single macroparticle and split it into two macroparticles, each with half the weight. This prevents regions of low physical density from becoming sparsely represented by macroparticles. Splitters are simple and work very well.

Combiners do the opposite of splitters, taking multiple macroparticles and combining them into a smaller number of macroparticles. This, however, is much more complex since the goal is not just to preserve the weight, but also the phase space information of the combined particles. There are therefore many algorithms to choose from within VSim. Descriptions of these algorithms can be found in the “nullSelfCombination” section in the Reference manual. There is, unfortunately, so a one-size-fits-all solution for combining, and often one must try multiple algorithms before finding the one that works best for a given simulation.

## 6.7.2 Combining Algorithms

Combining algorithms must consider which quantities to conserve through the combining process. Quantities to consider are mass, charge, energy, momentum, flux, and phase space distribution shape.

Algorithms:

- **PairwiseDecimation** - This is the simplest combination algorithm, which selects two particles, deletes one and adds its mass/charge to the other. While this will not conserve anything except for mass/energy for any given pair, the idea is that other quantities will be preserved on average. Assuming the decimation algorithm chooses particles uniformly randomly, the shape of the particle distribution should be unchanged, after many combinations, with the amplitude of the distribution decreasing. However, if the combining algorithm is slow compared to the physics processes, this will produce erroneous physics as processes will see intermediate distributions due to the combining.
- **PairwiseInelastic** - This algorithm randomly selects two particles and combines them into a single particle. This algorithm conserves mass, charge, and momentum (does not conserve energy). This is achieved by giving the resulting particle the mean position and velocity of the initial particles.
- **PairwiseInelasticLimited** - This algorithm is a pair-wise combining approach similar to **PairwiseInelastic** except that it allows users to avoid combining particles that are above the user specified limit. Just as in **PairwiseInelastic**, this algorithm conserves mass, charge, and momentum.
- **QuartetElastic** - This algorithm randomly selects four particles and combines them down to two. By selecting more particles, more degrees of freedom are available for conserving quantities. In this case, we conserve mass, charge, energy, and momentum.
- **PairwiseEnergyConserved** - This algorithm groups particles within a cell into velocity phase-space bins and then particles in each velocity phase-space bin are combined pair-wise. In this approach mass, charge, energy, and momentum are conserved, and only particles near each other in phase space are combined, attempting to also preserve the shape of the phase space.
- **FluxConserving** - This algorithm conserves mass, energy, momentum, and flux by combining four particles into two. This is ideal for simulations with particle beams or in general a net flow of particles (ie. non-zero mean phase-space distribution).

## 6.8 Histories

A *history* is an array of data that is output at every time step. The type of history that you can include is dependent on the type of simulation (i.e. particle, electrostatic, electromagnetic, etc.) that you are running.

You can incorporate a history in text-setup through one of two ways:

- Including a history by means of a *History* block
- Calling a macro that automatically generates history blocks for you based on a few select input parameters

More details and some example usage of these two methods will be provided later on in *Text Setup*. For further information on history types and parameters as a whole, you can also visit the “History” section in the Reference Manual.



## **ADVANCED SIMULATION TOPICS**

### **7.1 Selecting Solvers and Solver Parameters**

Solvers are used when the field is defined implicitly, i.e., when there is a relation between the field values at various locations and what one is given. For example, in an electrostatic simulation, the potential is found by solving Poisson's equation,

$$-\nabla^2\phi = \rho$$

This partial differential equation is then discretized to obtain a large linear equation that in the problem interior relates a linear combination of the values of the potentials at a node and nearby nodes to the value of the potential at that node. At the boundaries, the value of the potential is directly related to the desired boundary value. Consequently, Vorpal must solve a large linear system, where the number of independent values, i.e., the length of the vector, is the number of field values in the problem.

As an example, in a Poisson solve for a 10 cell by 10 cell problem, because the potential must also be known at the node above the last cell, there are  $11 \times 11 = 121$  values of potential to solve for. The matrix for this solve therefore has  $121 \times 121 = 14641$  elements. One can see that these matrices become very large as the problem size increases.

Vorpal gives the user a fair number of choices for solving these problems. The first choice is whether to use a direct solver, which uses methods like LU decomposition, or an iterative solver, which finds the solution through successive matrix operations that converge to the solution. This first choice generally depends on the size of the problem. For problems that are too large, one cannot hold the matrix in memory, and so one generally goes with an iterative method. This is not definitive, however, as for smaller problems, an iterative method may still get one to solution more rapidly.

When using a direct solver, it is important to ensure that the matrix is not singular. In almost all cases this is true, but with fully periodic boundary conditions, the Laplace matrix is singular, as can be seen by the fact that the constant function is in its null space. One can make the matrix nonsingular by applying a boundary condition at a single cell. However, this problem has no solution when the total charge in the system does not vanish. Thus, one must ensure that the system is overall neutral. This can be enforced in Vorpal by adding a neutralizing background charge density.

Iterative solvers have no problem with periodic boundary conditions, and with them one need not impose any single-cell boundary condition. Iterative solvers work very well on the Poisson problem. However, for iterative solvers one must choose a tolerance, which is a measure of the residual reduction compared to the initial residual, as the stopping criterion for the iterative solver. Values of  $10^{-8}$  often are sufficient for giving meaningful results. If the solver does not converge, increase the tolerance. If the resulting potentials miss a lot of small scale structure, reduce the tolerance.

For iterative solvers, one must also choose a preconditioner. Preconditioners transform the linear system used in the Poisson solver into systems with more favorable convergence behavior. For a simple fully periodic system, a preconditioner may not be necessary. For most other cases, using a pre-conditioner significantly improves the convergence behavior. Multi-grid preconditioning tends to yield the best convergence behavior and is especially good for Poisson's equation.

For a comparison of different pre-conditioners and solvers for electrostatic simulations, see P. Messmer et.al. [MB04].

The Visual Setup user interface by default sets up what are believed to be the best solvers, preconditioners, and parameters. However, it also allows the freedom to try others. Experimentation may be needed to arrive at the fastest running simulation.



## VISUAL SETUP TAB

### 8.1 Visual-setup Simulations

After you open a new or example simulation that is not text-based, GSimComposer displays the **Setup** window containing an *Editor* pane with a *Simulation Elements Tree*, a *Property Editor*, and a *Geometry View* to allow for easy creation of your simulation. The icon panel remains available on the far left.

**Note:** A *Navigation Pane* can be shown by clicking and dragging on the vertical bar separating the *Icon panel* from the *Editor* pane. See Fig. 8.1.

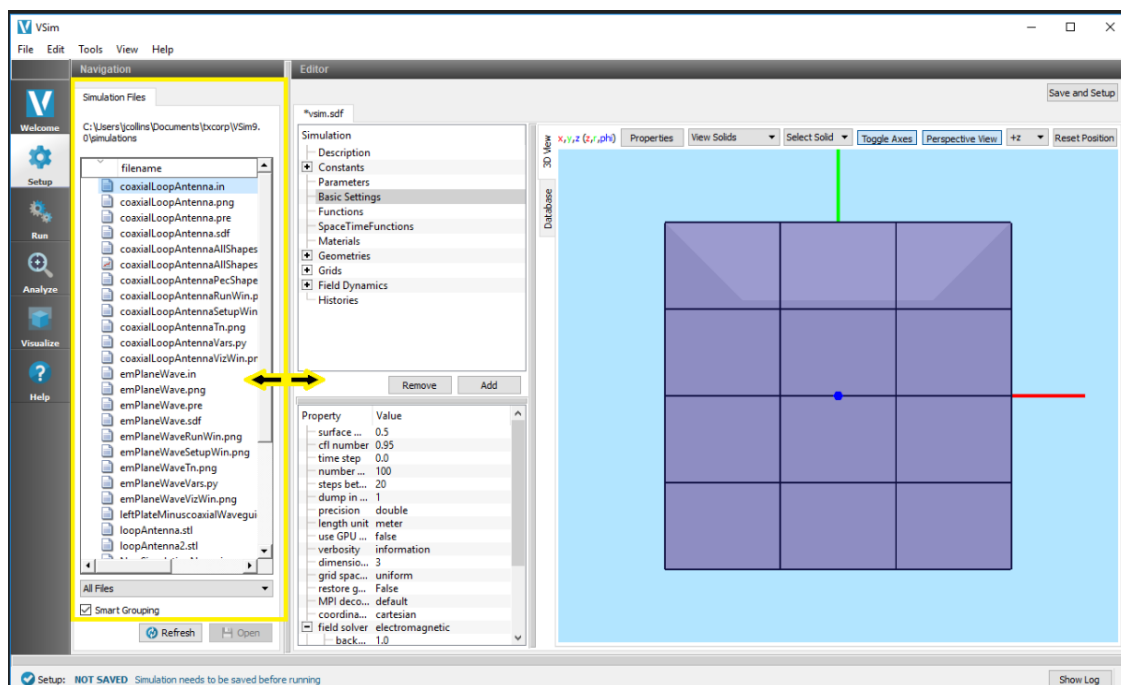


Fig. 8.1: The Navigation Pane

The following sections will go through each of the components of the **Setup** window.

For in depth information on each of the properties and possible values outlined or described in the rest of the chapter, please see the “Composer Basic Settings” in the Reference manual.

The figure *Visual based setup window* illustrates the layout of the GSimComposer **Setup** window using labels for the parts of the interface to which this introduction and the tutorials refer. See Fig. 8.2.

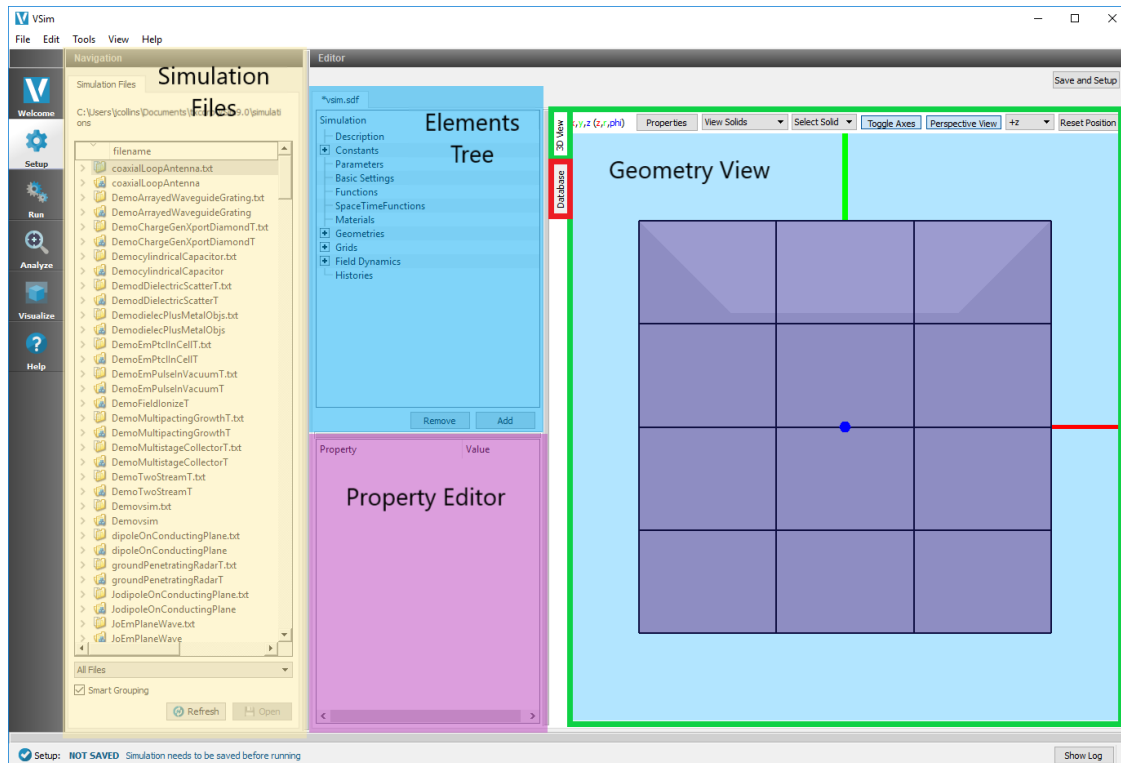


Fig. 8.2: Visual based setup window

## 8.2 Navigation Pane and Simulation Files

The *Navigation* pane contains a list of *Simulation Files*.

To enable convenient viewing of the list of simulation files, GSimComposer allows you to specify in what order as well as which type(s) of files you would like to view. *Smart Grouping* causes similar types of files to be displayed in the same area of the *Simulation Files* tab list. Turning off *Smart Grouping* causes files to be displayed in alphabetical order rather than by type. *All Files* indicates that you want to see all available files involved in the simulation. You could choose to limit your view to only *Simulation Files*, which are files such as input files and macros that can be edited in the GSimComposer Editor pane, or *Text* files, which include all types of human-readable file formats, or *Data* files, which include incremental dump files and output files that can be visualized.

## 8.3 Elements Tree Overview

You can navigate through the *Elements Tree* using either your mouse or your keyboard arrows. The up and down arrows will scroll up and down through the list of elements, while the right and left arrows will respectively expand and collapse the elements. To double click, press F2.

Highlighting a particular element in the tree will cause the *Property Editor* to update with Property/Value pairs that can be edited.

New elements can be added under an element in the tree by **Right Clicking**. This will be context sensitive, so for example right clicking *Particle Sources* will open up a tree of all types of available particle sources to choose from. Select one and click on it to add a new instance of that element.

### 8.3.1 Turn Off/On

This is an option available after **Right Clicking** or using the **Add** button. If it is selected it will gray out the element and it will not be translated into the input file.

This can be particularly useful for comparing results with and without a single element in place without having to go through the process of completely re-specifying it.

### 8.3.2 Create Clones

Any object can be right clicked and cloned, with all settings set the same as the object. Any number of copies can be created.

### 8.3.3 Buttons

- Undo

This will Undo the last thing done. For example it will Undo the addition of an element, or the setting of a property.

- Add Multiple

This is a shortcut that will add another instance of the last element added to the tree. Note that this can only be used on elements that are not visualized in the 3D view.

- Remove

This will removed the highlighted element of the tree.

- Add

This is the equivalent of right clicking a selected Element in the tree, and will present the same options.

We now discuss each of the sections of the Elements Tree.

## 8.4 Description

The *Description* element holds basic user-supplied text information about the simulation.

## 8.5 Constants

The *Constants* element contains a set of pre-defined physical constants that can be used in other elements of the simulation. You may also define your own by highlighting *Constants* and either clicking the *Add* button at the lower right of the *Elements Tree* or right-clicking and selecting *Add Constant* → *User Defined*.

The name of the user-defined *Constant* can be modified by double clicking on the element in the *Elements Tree* and typing in a new name.

**Note:** A constant name can contain only alphanumeric characters and underscore and must start with a letter. Our convention is to use ALL\_CAPS for constants.

A value can be given to the user defined *Constant* by double clicking on the value in the *Property Editor*. See Fig. 8.3.

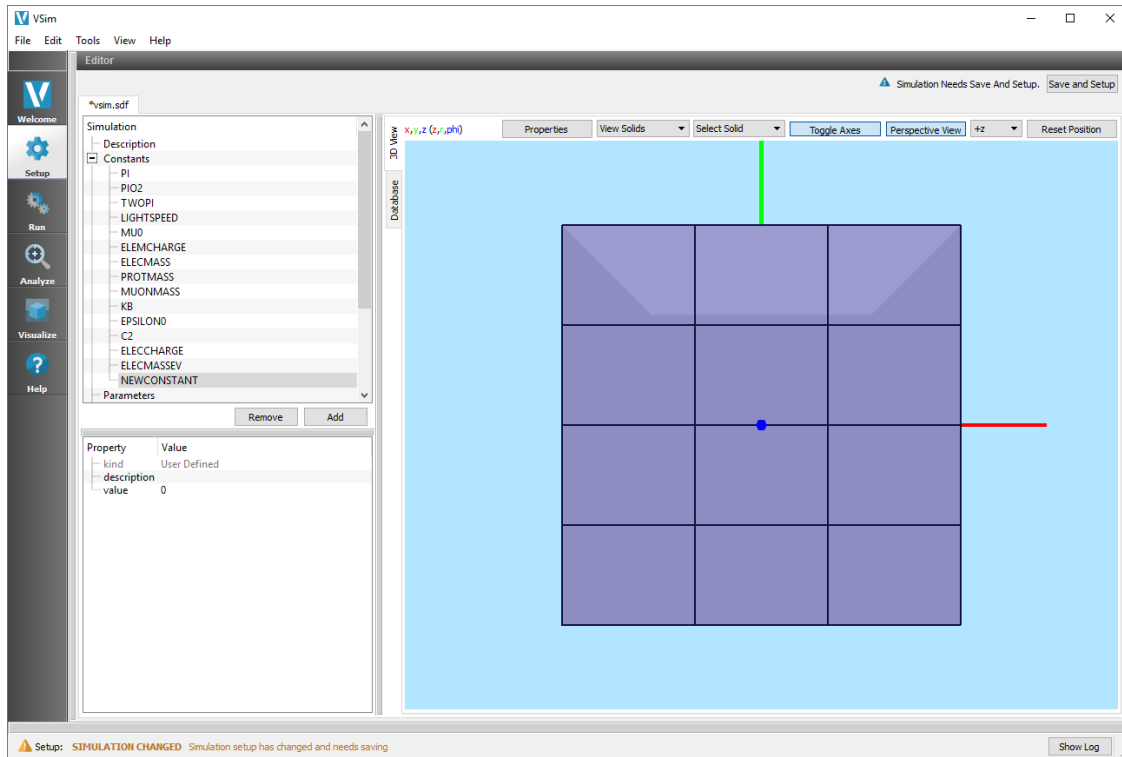


Fig. 8.3: Constant definitions

## 8.6 Parameters

The *Parameters* element is a location for *evaluated*, user-defined, variables that can be used in other elements of the simulation.

You can add a *parameter* by highlighting *Parameters* and either clicking the *Add* button at the lower right of the *Elements Tree* or right-clicking and selecting *Add Parameter -> User Defined*.

The name of the user-defined *Parameter* can be modified by double clicking on the element in the *Elements Tree* and typing in a new name.

**Note:** A parameter name can contain only alphanumeric characters and underscore and must start with a letter. Our convention is to use ALL\_CAPS for parameters.

An expression can be given to the user defined *Parameter* by double clicking on the expression value in the *Property Editor*. You can use any of the *Constants* as well as any real number in the expression. See Fig. 8.4.

**Note:** If the expression is not valid, the *parameter* will appear in red in the *Elements Tree*.

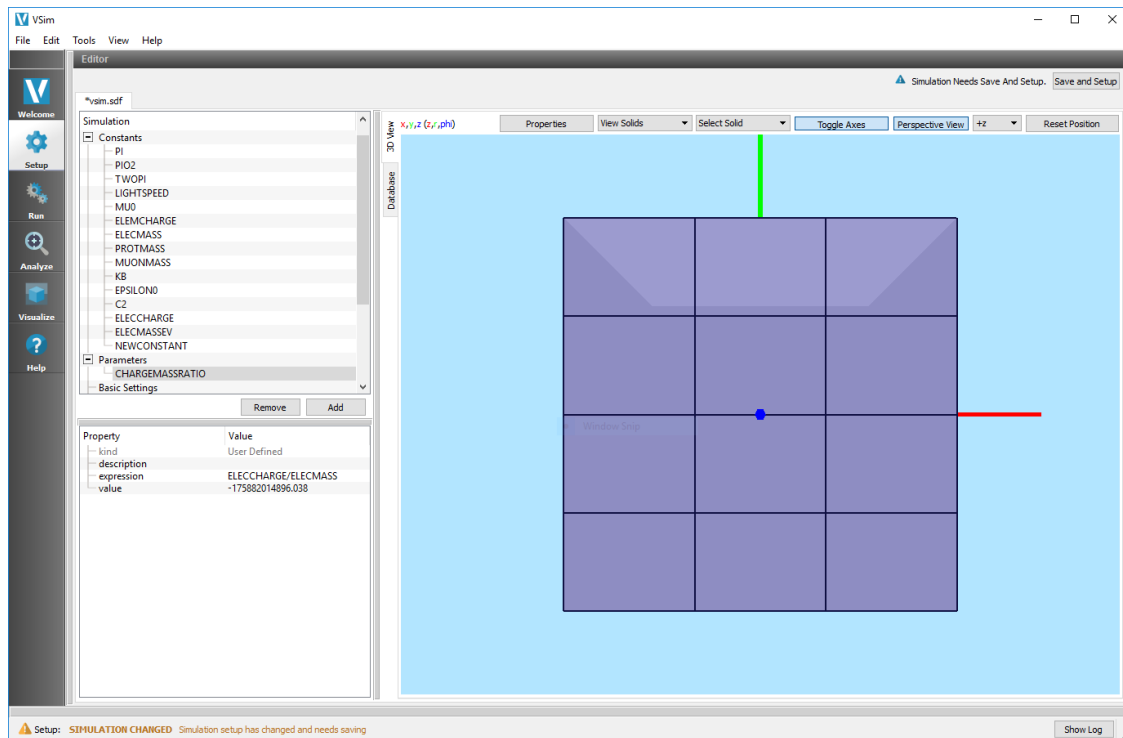


Fig. 8.4: Parameter definitions

## 8.7 Basic Settings

The *Basic Settings* element contains a group of property/value pairs that define the basic setup of the simulation.

Here you can find properties such as the type of field solve (electromagnetic, electrostatic, prescribed field, constant fields, hybrid fluid, or no field), the dimensionality (3D, 2D, 1D), whether or not to include kinetic particles in the simulation, and the time step.

## 8.8 Functions

The *Functions* element is a location for writing user-defined functions that can be used in simplifying the definition of a *SpaceTimeFunction*. The function can contain any number of arbitrary arguments and is not limited to the default values of *x* and *y*.

**Note:** A *Function* can be used to define another *Function* or a *SpaceTimeFunction*. *Functions* cannot be used to define an expression in other elements nor can they be used to define parameters. Expressions are defined by *SpaceTimeFunctions*.

To create your own function, highlight *Function* and either click the *Add* button at the lower right of the *Elements Tree* or right-click and select *Add Function* → *User Defined*.

The name of the user-defined *Function* can be modified by double clicking on the element in the *Elements Tree* and typing in a new name.

**Note:** A function name can contain only alphanumeric characters and underscore and must start with a letter. Our convention is to use lowerCamelCase for function names.

You can define the *Function* by double clicking on the expression value in the *Property Editor*. You can use any of the *Constants*, *Parameters*, or *Functions* previously defined, as well as any real number or Python operator in the expression. See Fig. 8.5.

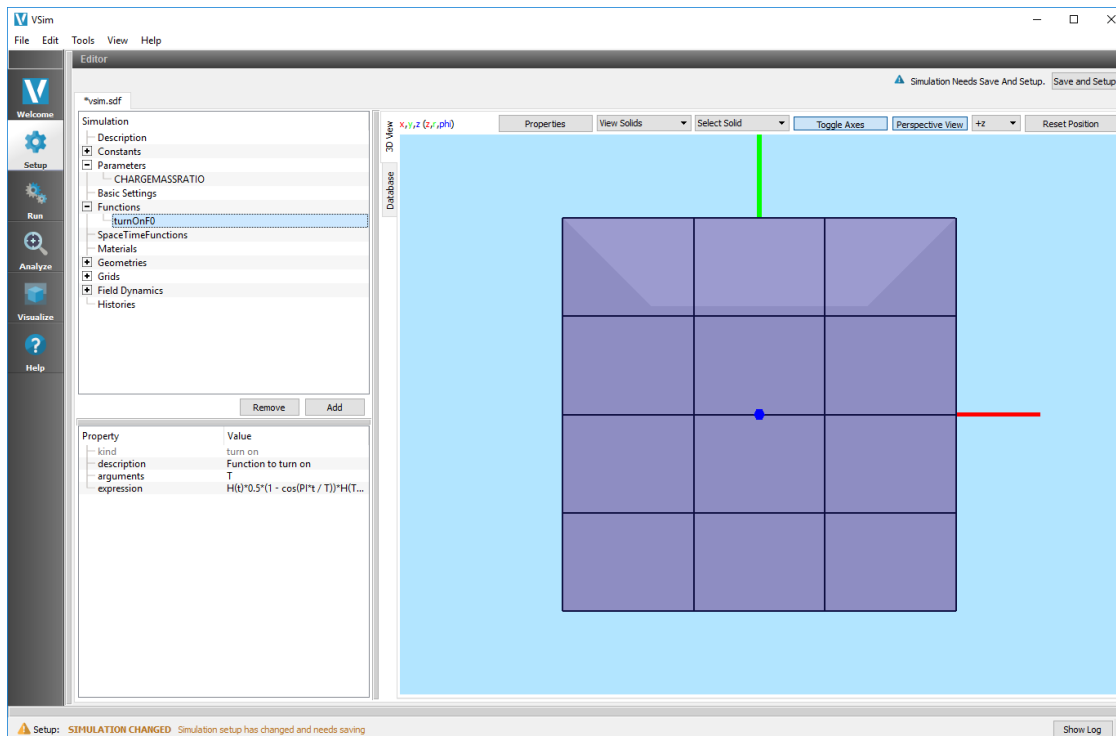


Fig. 8.5: Function definitions

## 8.9 SpaceTimeFunctions

The *SpaceTimeFunctions* element is a location for writing user-defined functions that specifically depend on the spatial and temporal variables  $x$ ,  $y$ ,  $z$ , and  $t$ . A *SpaceTimeFunction* can be used in other elements of the simulation by right clicking on the value and selecting the defined *SpaceTimeFunction* as shown in the figure *SpaceTimeFunctions definitions*.

To create your own function, highlight *SpaceTimeFunction* and either click the *Add* button at the lower right of the *Elements Tree* or right-click and select *Add SpaceTimeFunction* → *User Defined*.

The name of the user-defined *SpaceTimeFunction* can be modified by double clicking on the element in the *Elements Tree* and typing in a new name.

**Note:** A *SpaceTimeFunction* name can contain only alphanumeric characters and underscore and must start with a letter. Our convention is to use lowerCamelCase for *SpaceTimeFunction* names.

You can define the *SpaceTimeFunction* by double clicking on the expression value in the *Property Editor*. You can use any of the *Constants*, *Parameters*, or *Functions* defined above, as well as any real number or Python operator in the expression. See Fig. 8.6.

**Note:** With SpaceTimeFunctions you may type in the name of a *Constant* or *Parameter* directly, however if you are to change the name of the Constant / Parameter used it will not automatically update to the new name, as it will in other elements of the SDF file.

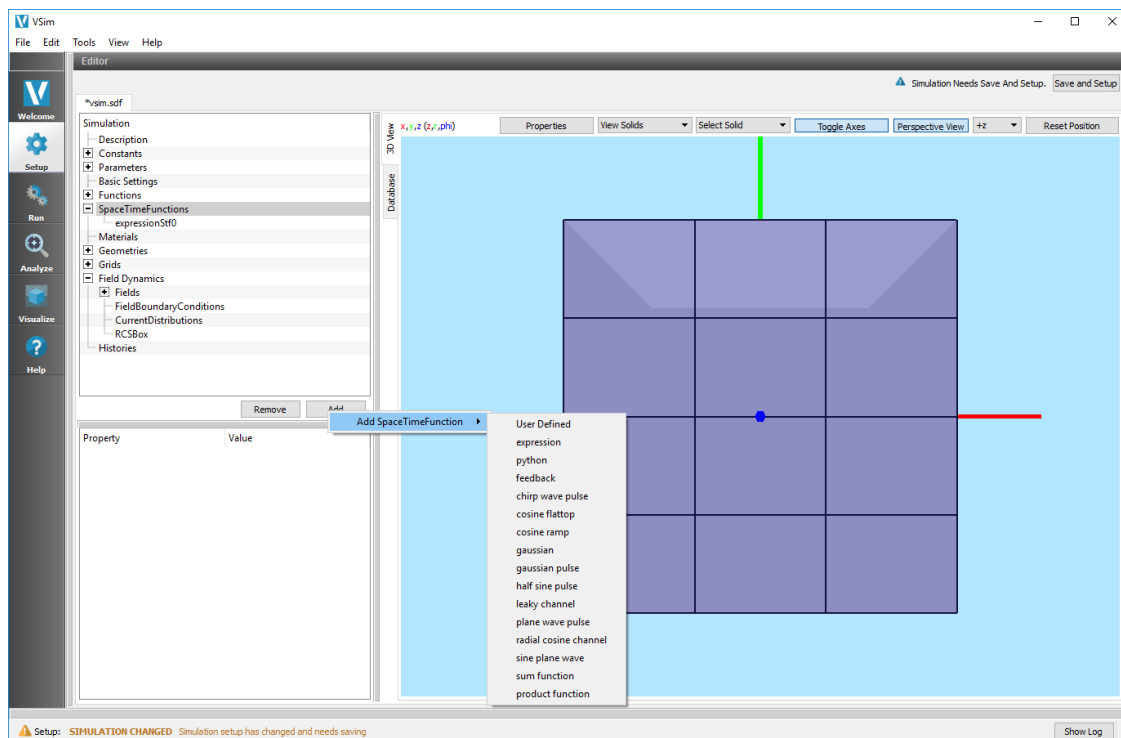


Fig. 8.6: SpaceTimeFunctions definitions

## 8.10 Materials

The *Materials* element holds information about any materials used in the simulation. There are some *Materials* built into Vsim, and the user may import other desired materials.

To import a *Material*, either click the *Add* button at the lower right of the *Elements Tree* or right-click and select *Import Materials*.

The *Materials* file must have the extension *.vmat*. You can specify your own materials file to import special materials specific to your simulation. See Fig. 8.7.

**Note:** Tech-X has a standard materials file distributed with Vsim. It is located in the data folder of your installation.

Once a material file has been imported into GSimComposer, you can add a specific material to your simulation by highlighting the material of choice and clicking on the *Add To Simulation* button.

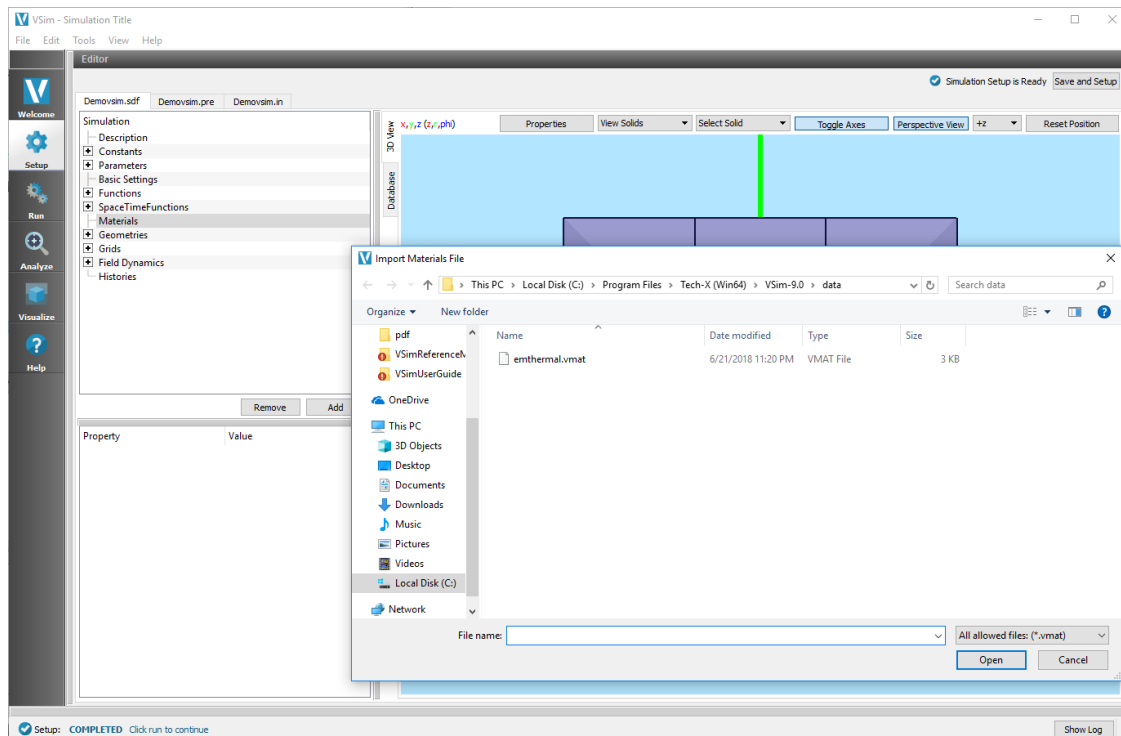


Fig. 8.7: Importing materials

After a material is in the simulation, you can see it under the *Materials* element. The material properties can be modified, if desired. The *Materials* can be assigned to a geometry in the *material* property in the *Properties Editor* pane. See Fig. 8.8.

## 8.11 Geometries

The *Geometries* element contains information about any geometries that are in the simulation. You can import a file, or create your own with CSG. Geometries present in the simulation can be modified, for example, healed, with holes closed.

Expanding the *Geometries* sub-elements view will show the individual parts (if any) of the imported geometry, or CSG built geometry.

To hide a specific part, uncheck the box next to it.

---

**Note:** Hiding a part of the geometry will not remove it from the simulation. Vsim will use the full geometry defined in the imported file.

---

For use in the simulation, a *Geometries* part **MUST** have a material assigned to it, other wise it is ignored (treated as vacuum).

The material can be assigned to a geometry by double clicking on the material value in the *Properties Editor*. Note that the value of the dielectric assigned to the material, and hence the geometry, is only used if the field solve is electromagnetic. If you are using an electrostatic field solve, the dielectric must be assigned using *SpaceTimeFunctions*. Two examples illustrate the difference in how Vsim handles dielectrics in the electromagnetic and electrostatic field



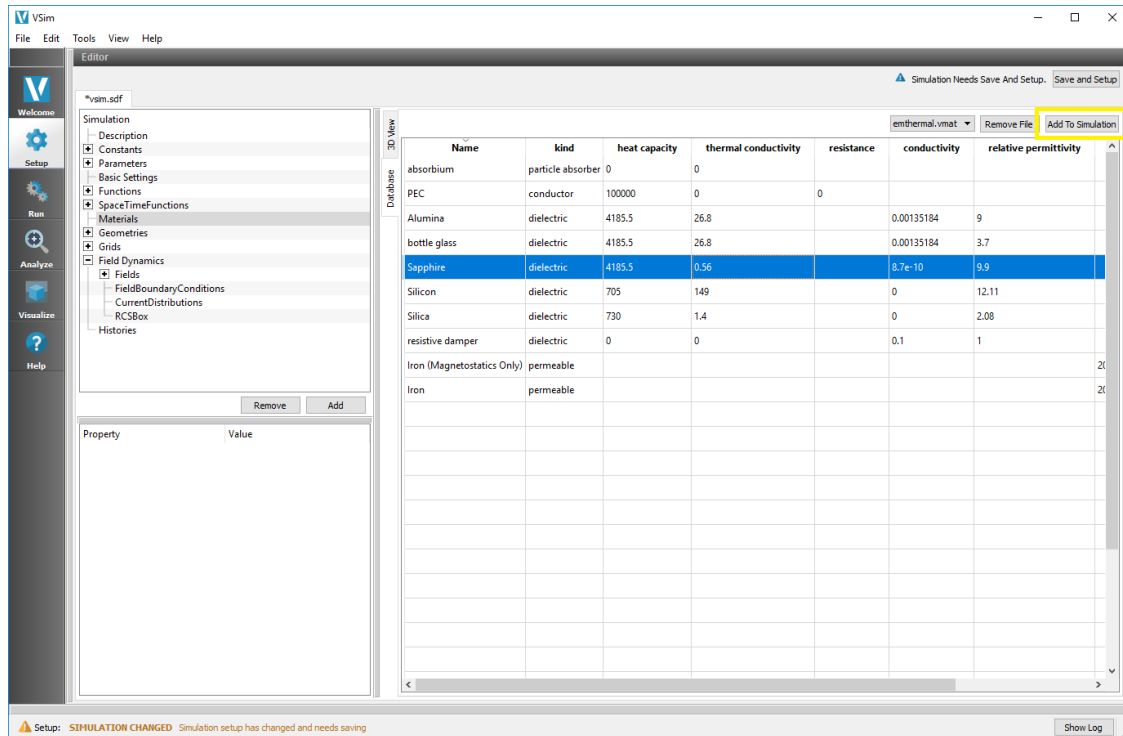


Fig. 8.8: Adding materials to the simulation.

solves: “Dielectric in Electromagnetics” and “Dielectric in Electrostatics” both of which are part of the “Vsim for Electromagnetics” examples.

The color of geometry can be set in the *Properties Editor* after selecting the item line of that geometry in the tree. Not that selecting an item is different that selecting the visibility checkbox of that geometry item. Selecting the item means clicking on the line of the item anywhere but at checkbox. Once an geometry item is selected a color property line should appear in the *Properties Editor*. Double click on the color box in the *Value* column to bring up a *Select Color* dialog window to set the color. The transparency control is “alpha” on Windows and Linux (see Fig. 8.9) and “opacity” on Mac (see Fig. 8.10) can also be set in the *Select Color* dialog.

### 8.11.1 Importing a Pre-defined Geometry

To import a geometry into your simulation, highlight the *Geometries* element in the *Elements Tree* and click on the *Add → Import Geometries* button located at the bottom of the *Elements Tree*, or simply right click on the *Geometries* element → *Import Geometries*.

Here you can navigate to a supported file type and open the file. Supported file types include:

- Step Files (.stp, .step, .p12)
- STereoLithography Files (.stl)
- Visualization Toolkit Files (.vtk)
- Polygon File Format (.ply)

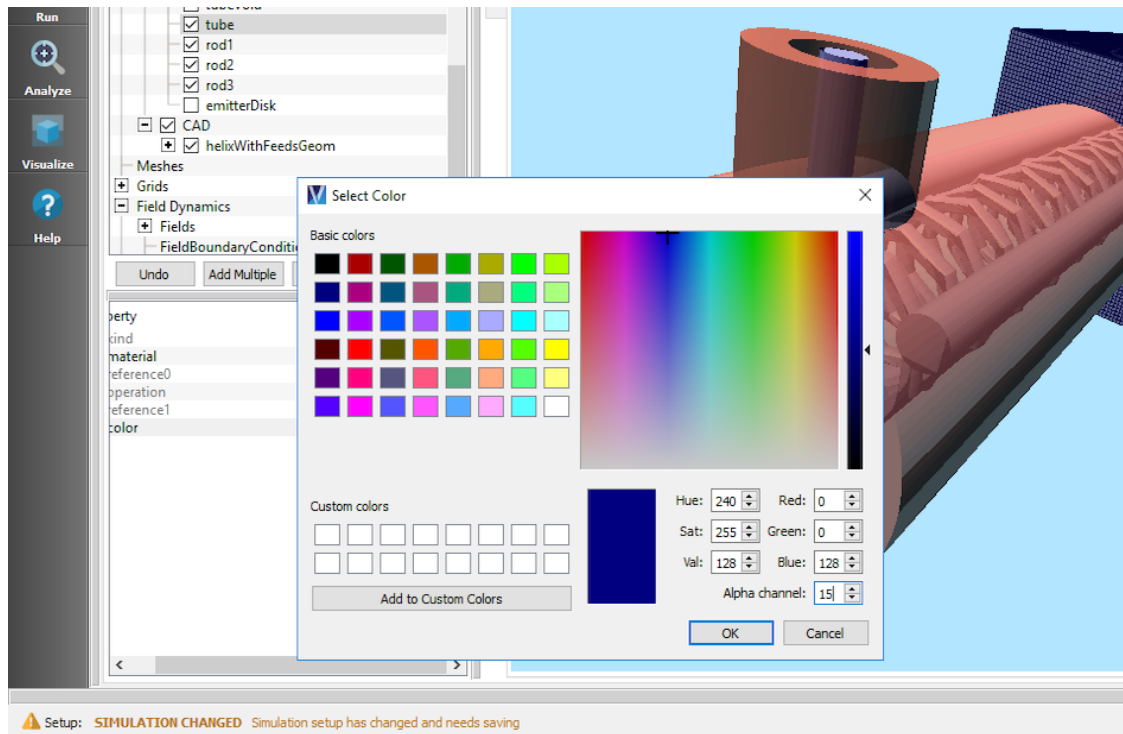


Fig. 8.9: Setting the color property of a geometry through the Select Color dialog on Windows and Linux. The alpha controls the transparency of the selected geometry.

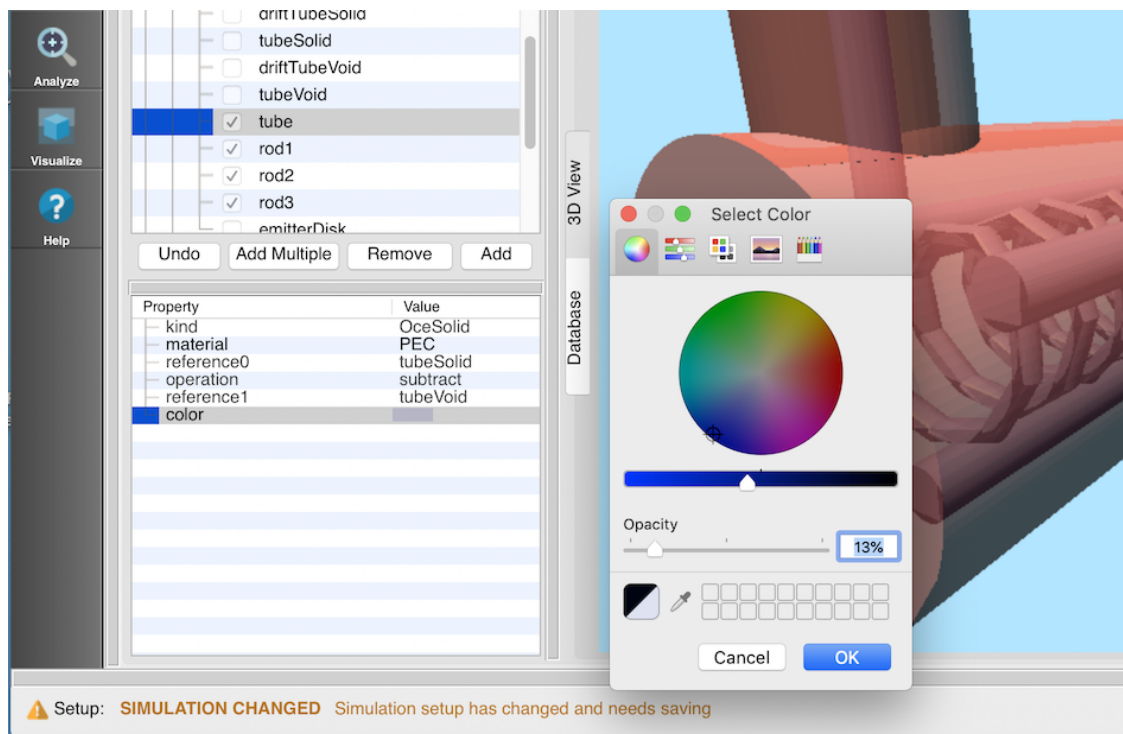


Fig. 8.10: Setting the color property of a geometry through the Select Color dialog on Mac. The opacity controls the transparency of the selected geometry.

## 8.11.2 Building Your Own Geometry

You can build your own geometry using Constructive Solid Geometry (CSG) to create a complex shape by combining simple shapes using boolean operators.

To do this, highlight the *CSG* element and right click *Add Primitive* and select one of the pre-defined shapes. See Fig. 8.11.

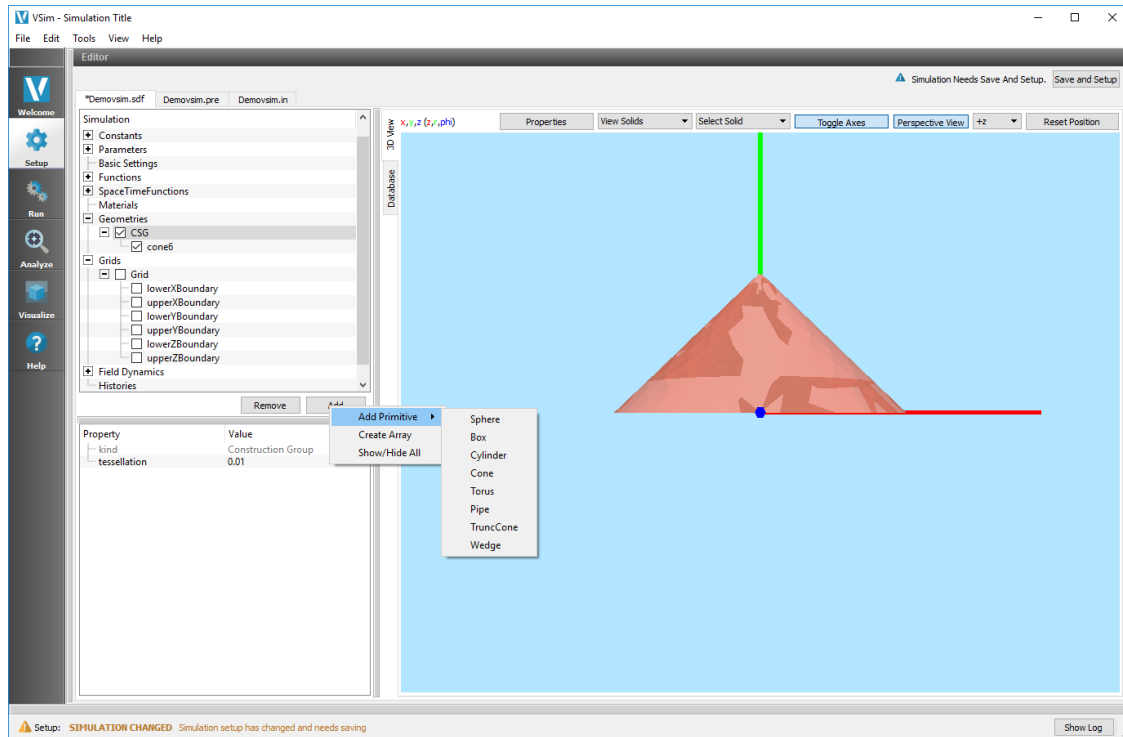


Fig. 8.11: Constructive Solid Geometry (CSG)

After multiple CSG shapes have been added, you can either subtract, union, or intersect them with a boolean operation. This will create a new *Geometries* element.

To do this, highlight the number of shapes you would like to include in the boolean operation, right click, and select the boolean operation you want. See Fig. 8.12. To highlight the second shape on Windows, hold down the 'control' button before selecting the second shape.

---

**Note:** The order of highlighting your shapes matters when doing the subtract boolean operation. The second shape will be subtracted from the first. The boolean operation menu will show the operation to be performed based on the order of highlighting.

---



---

**Note:** If a shape is not part of a combined shape through a boolean operation, you can assign a material to it. Once a shape has been combined with another shape, only the combined shape may be assigned a material.

---

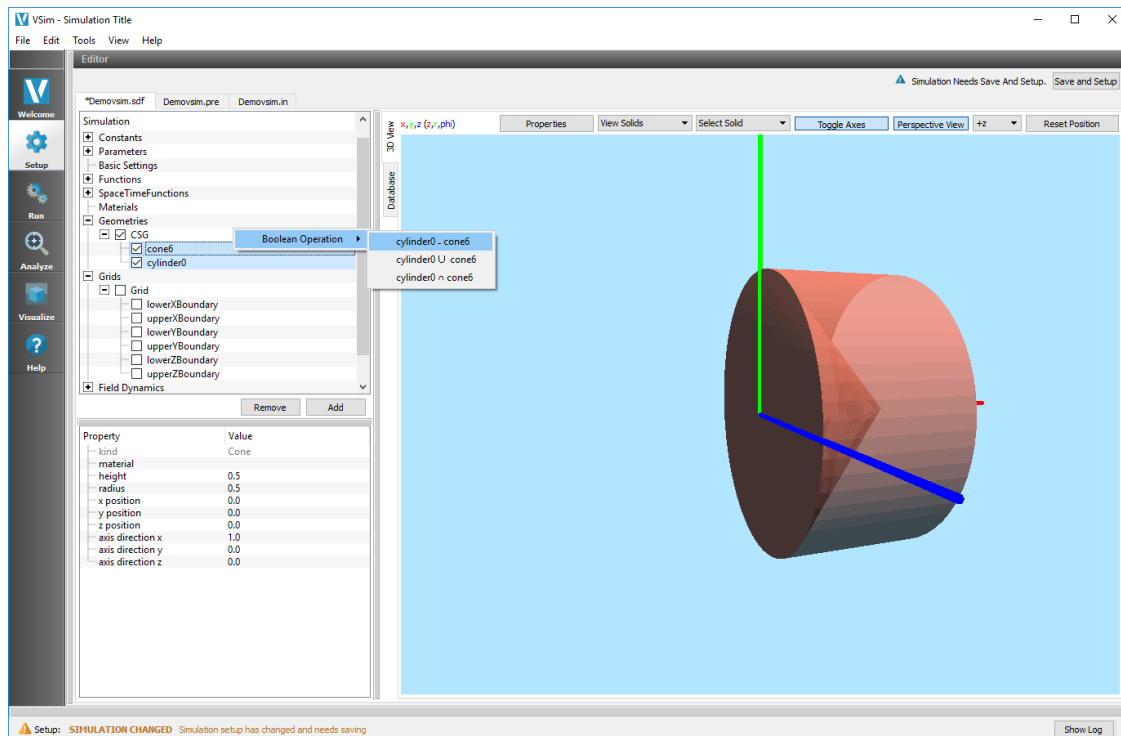


Fig. 8.12: Constructive Solid Geometry (CSG) Boolean Operator

### 8.11.3 Array Operations

Both CSG and CAD objects may be duplicated with array operations. To do this right click the object in question, and select **Create Array**. This will then open a dialog window to select the *X Count*, *Y Count*, and *Z Count* which will specify the number of objects to create in the respective axial direction. A drop down menu is also available to specify this number with a Constant or Parameter. *X Spacing*, *Y Spacing* and *Z Spacing* will control the spacing between objects - this spacing is in reference to the object in question. Similarly it is possible to assign a Constant or Parameter using the same drop down menu.

There is a Check box available to *Union Created Shapes* which will perform a union boolean operation between all shapes.

This dialog box is shown in the image below

---

**Note:** If using a constant or parameter, to define number of array elements/spacing it will hardcode to the value of that constant/parameter at the time of array creation. It will not update the number of elements/spacing should the value of that constant/parameter later change.

---

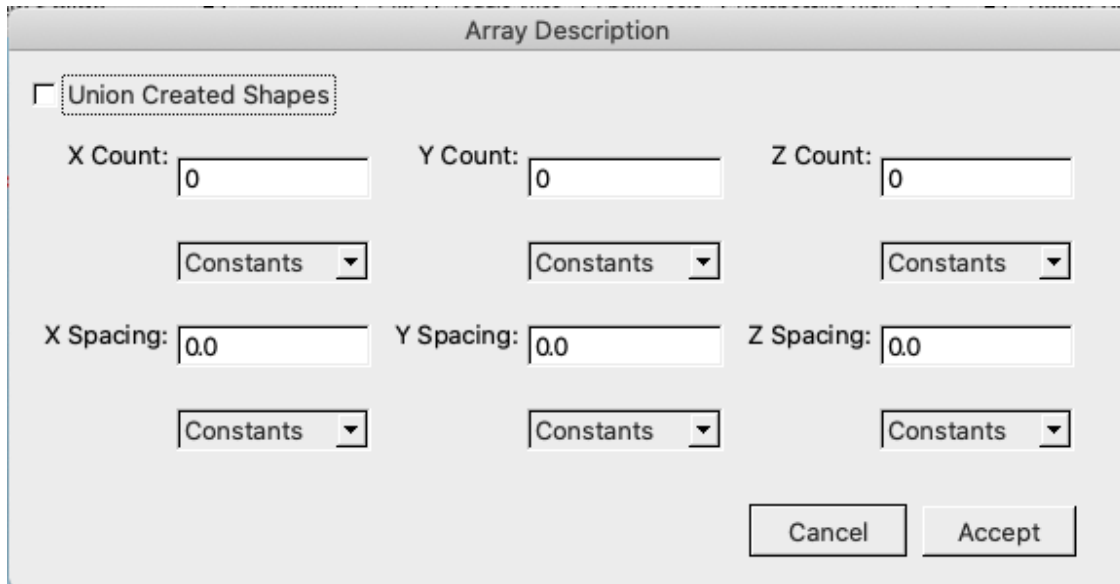


Fig. 8.13: Array Creation

#### 8.11.4 Creating Triangulated Surfaces

One can perform many other operations on surfaces in this interface, including analysis of geometries, which involves determining the number of separate surfaces that a geometry contains, and healing of geometries, which involves making them water-tight, closing up any holes. to triangulated form. One can also generate the simulation mesh for the geometries, and visualize it, which allows one to see whether the geometries have fidelity sufficient for a quality simulation.

#### Separating geometries

A common flaw is a CAD file that contains many objects, only some of which are required for the simulation. These objects may also interfere with one another, and may obstruct viewing angles. To begin, one imports the geometry to visually inspect it and see what it contains. One may then use the *Separate Surfaces* command to identify individual objects in the file and extract them into separate files for further use. This is invoked through the menu shown in Fig. 8.14, which comes up through a right click on the geometry.

Selecting *Separate Surfaces* brings up the window shown in Fig. 8.15. Through this interface, one can separate the surfaces. Ultimately, one saves the surfaces for use as separate geometrical items in the simulation. Regardless of what options are chosen in this dialog, the original geometry object will not be changed.

The first step is to “Analyze” the geometry, which will calculate the connectivity between each of the polygons that make up the geometry (triangles, quads, etc). By default, this step will not resolve any ambiguous connectivity - any polygon that could be a part of multiple geometries will be treated as a separate surface. To allow this step to attempt to resolve these ambiguities, one may turn on “Force maximum connectivity”. This step may take some time if the geometry is large or particularly complex.

The second step is to identify what types of surfaces to extract. One may choose to extract only closed surfaces, only open surfaces, or both. One may also set a size range for the extracted geometries. Only objects that satisfy all of the given conditions will be extracted.

The final step is to choose how the extracted surfaces will be saved. By default, each surface will be extracted into a new geometry object. You may also choose to group all geometries with the same size, or with the same type, or you may choose to save all surfaces into a single new geometry object. You may also supply an optional suffix to be

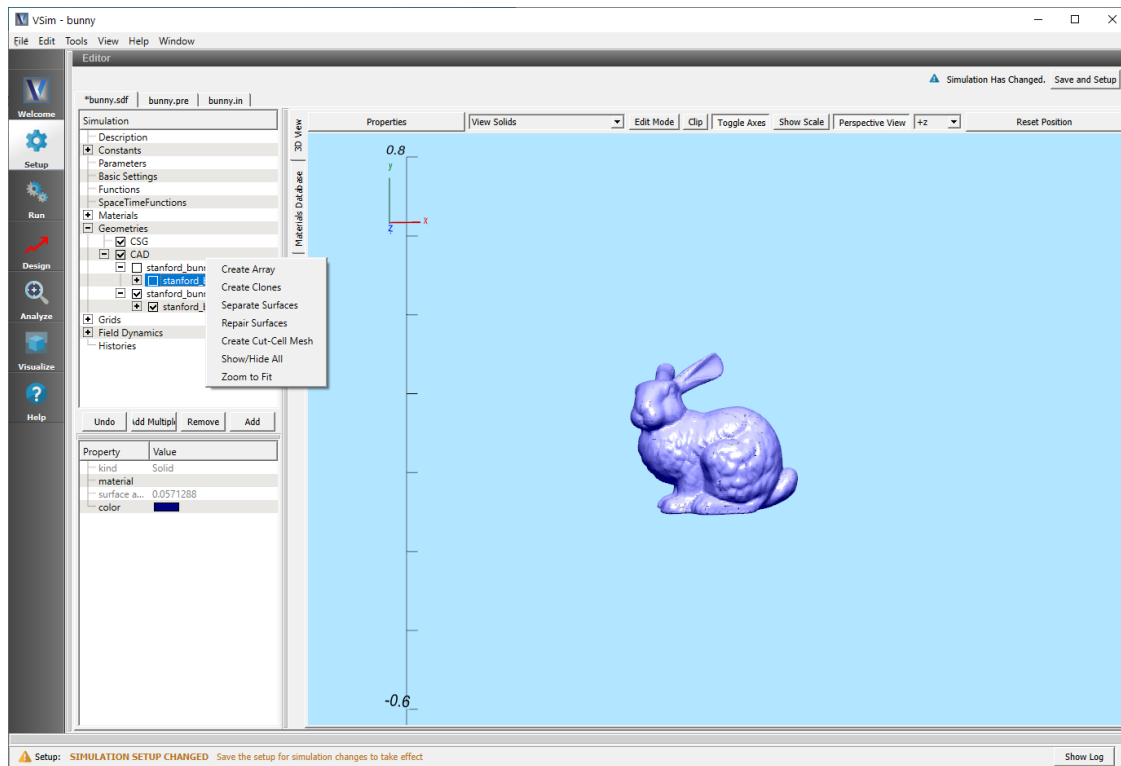


Fig. 8.14: Geometry analysis menu.

appended to each of the extracted geometries. Upon being saved, the separate objects will show up in the Geometries tree.

For complex geometries, it may be desirable to use *Separate Surfaces* multiple times. For example, one may first choose to extract all geometries that are “open”, i.e. geometries that must be repaired before use. One may then choose to extract only the largest of these geometries, discarding any objects that consist of only one or two triangles. Finally, one may save each of these large geometries into separate files for further analysis and repair.

## Healing geometries

Imported geometries may also have damage, such as holes, that will negatively affect the simulation. To detect and repair such geometry issues, one may use the *Repair Surfaces* command shown in Fig. 8.14. A dialog will appear that will guide you through the process. Again, regardless of what options are chosen in this dialog, the original geometry object will not be changed.

The first step is to “Analyze” the geometry, which will calculate the connectivity between each of the polygons that make up the geometry (triangles, quads, etc). By default, this step will not resolve any ambiguous connectivity - any polygon that could be a part of multiple geometries will be treated as a separate surface. To allow this step to attempt to resolve these ambiguities, one may turn on “Force maximum connectivity”. This step may take some time if the geometry is large or particularly complex.

When analysis has completed, the dialog will display a list of all holes (if any) in the target geometry. The “Edge Count” value is the number of edges that border the hole. A hole with three edges is a triangle, four edges is a quad, and more edges that that is a larger polygon. The “Display” button will rotate and zoom the view in an attempt to present the given hole to the camera. Note that this may not be successful if the hole is topologically complex or hidden by another surface. The “Repair” button allows you to fill the hole with new triangles. You may then also undo the operation if the repair is unsatisfactory. Upon conclusion of Repair, the patch needed for the repair shows up in the tree.

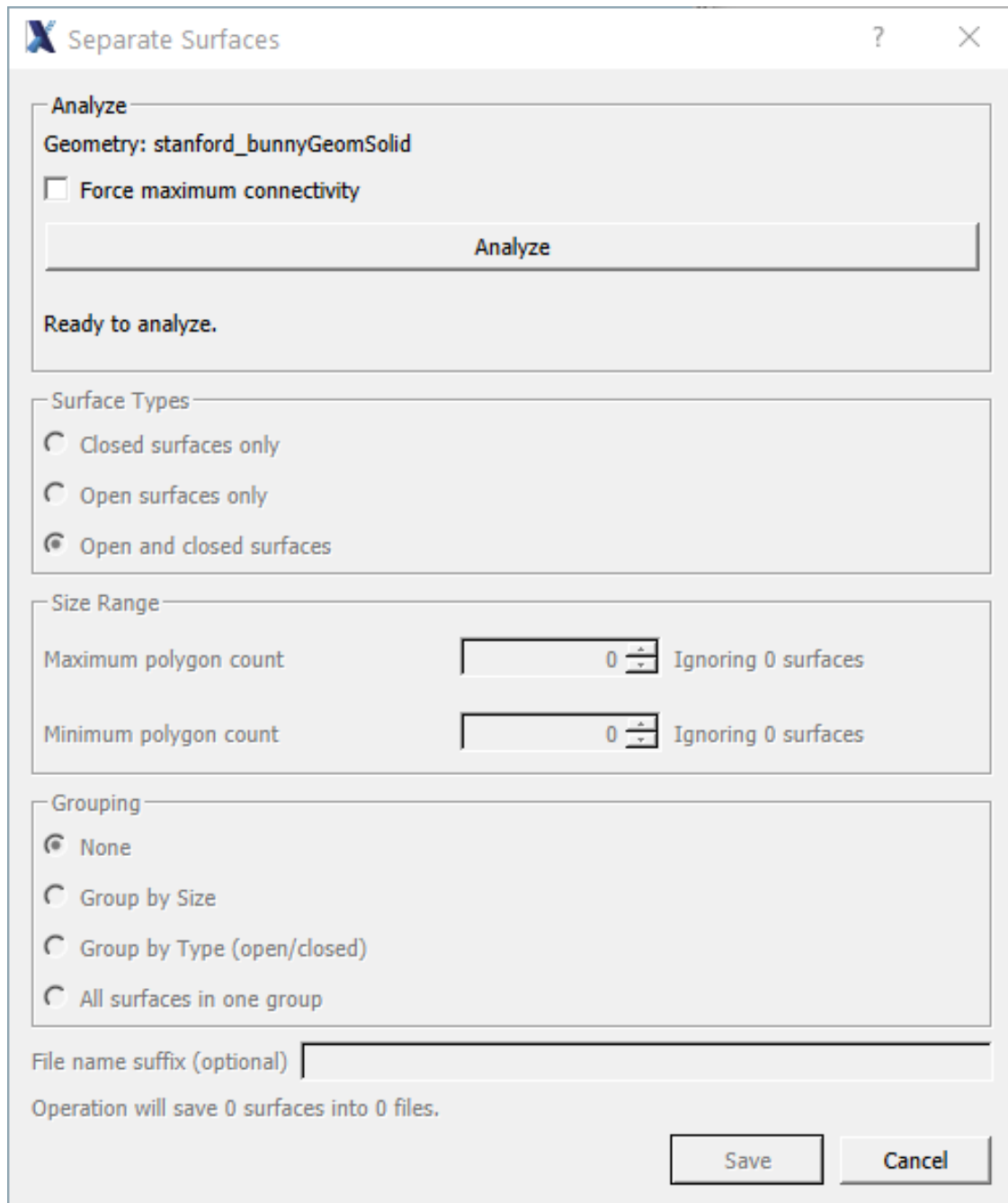


Fig. 8.15: Window for separating surfaces.

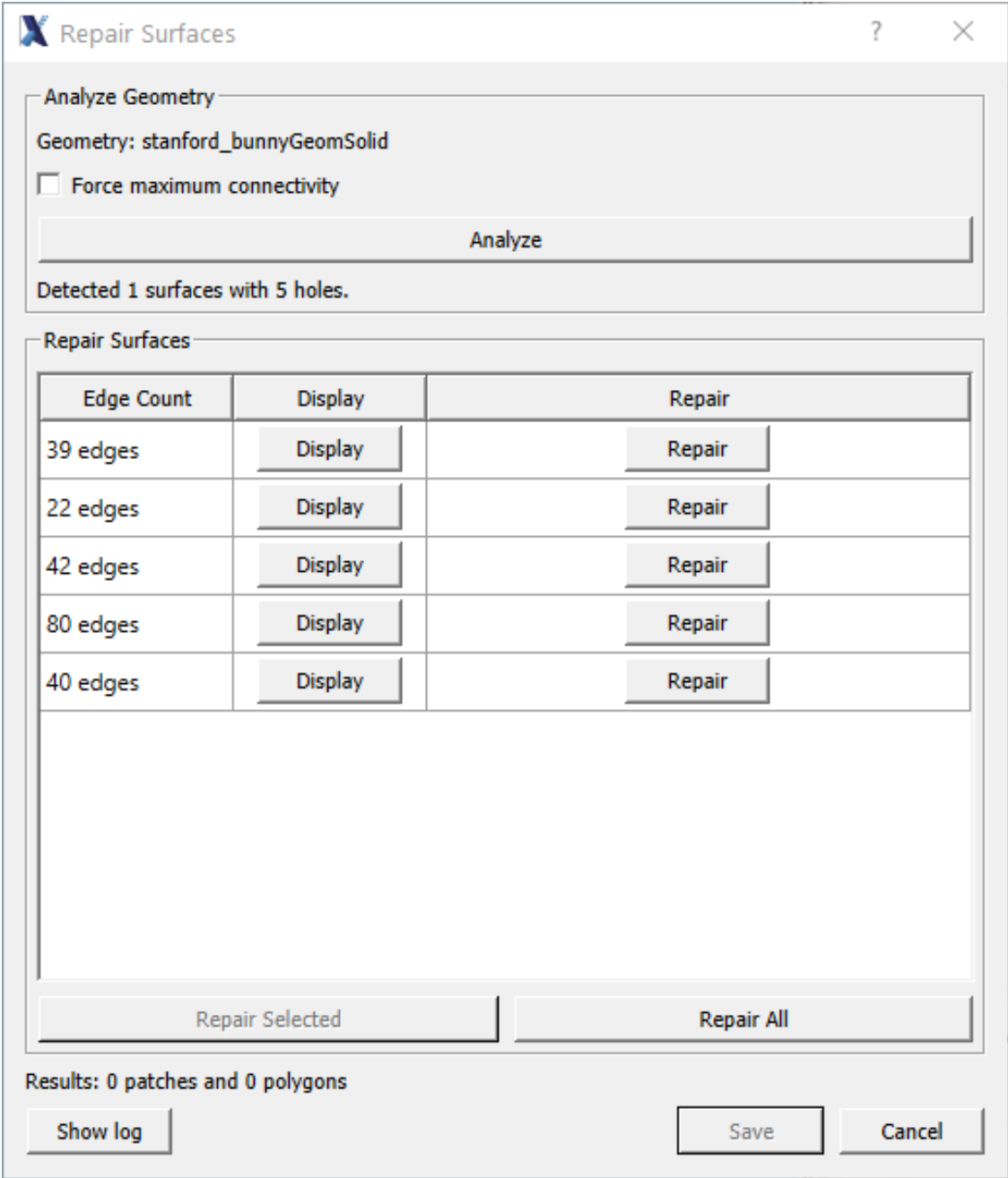


Fig. 8.16: Window for repair surfaces.



Composer includes several different healing algorithms. By default, the choice of algorithm is performed automatically depending on the characteristics of each hole. However, one may override the automatic choice if the resulting patches are not acceptable.

To finalize the process, press the “Save” button. Another dialog will appear that will allow you to choose how to save the results. “Export Healed Geometry” will create a new geometry object that includes the original geometry and any newly generated triangles. “Export Patch” will create a new geometry object that contains only the newly generated triangles. The healed object will then show up in the tree for use in the simulation.

### Re-calculating triangles

Many common geometry file formats (such as STL) contain a collection of triangles that represent the surface of the geometry. Depending on the original source of the file, the triangles may not be perfectly suited to the needs of the simulation. A geometry can be overly complicated, which results in longer compute times for geometry operations and increased file sizes; or it may be too simple, which interferes with transformations such as adding roughness.

The “Retriangulate Surfaces” command allows the number of triangles in a geometry to be changed to match the needs of the simulation. This command is available in the right-click menu on most types of imported geometry files, listed in the CAD section of the simulation tree view.

“Retriangulate Surfaces” begins by analyzing the geometry to calculate the connectivity between each of the triangles in the object. Next, it allows the user to choose one of two possible operations:

#### 8.11.5 Reduce triangle count

This operation will reduce the number of triangles in the surface by merging neighboring triangles that are sufficiently similar in . The Removal Percentage is the target amount of triangles to remove from the surface. Normal Tolerance provides a quality metric by which neighboring triangles are merged - if two triangles have normal vectors that differ by less than this value, the triangles will be merged. Finally, the option to Lock Vertexes allows the boundary edges of a non-solid geometry to be kept in their current location, and only interior triangles will be removed.

#### 8.11.6 Increase triangle count

This operation will increase the number of triangles in the surface, subject to a desired maximum edge length. Any triangle with an edge longer than the maximum desired value will be subdivided. Note that even small changes of the desired maximum edge length can result in dramatically larger number of triangles, so it’s best to approach the desired value from the high side.

### Generating Simulation (Cut-Cell) Meshes

Once a shape has been created using a primitive or imported, the shape will appear under the “CSG” tab if you created the shape or “CAD” tab if you imported the shape. If you want to view the mesh that is used to represent that shape in the simulation, then right click on the shape and click on *Create Cut-Cell Mesh*. The mesh will be added under the “Meshes” tab in the Visualization pane.. If you uncheck the box next to the shape under the “Geometries” tab, this will hide the solid shape, revealing the surface meshing used in the simulation. If the mesh is not revealed, check the box next to the shape under the “Meshes” tab. The resolution of the surface mesh depends on the resolution of the grid, which is determined under the “Grids” tab. Increasing “xCells” makes both the grid and surface mesh finer in the x-direction.

## 8.12 Grids

The type of grid is determined in the *Basic Settings* element. Its parameters are determined in the *Grid* element.

By default, a uniform Cartesian grid is added to your simulation with dimensions of 1m x 1m x 1m and cell numbers of 3, 4, and 5 in x, y, and z respectively.

To modify the type of grid, change the *Basic Settings* properties *coordinate system*, *dimensionality*, and *grid spacing*. See Fig. 8.17.

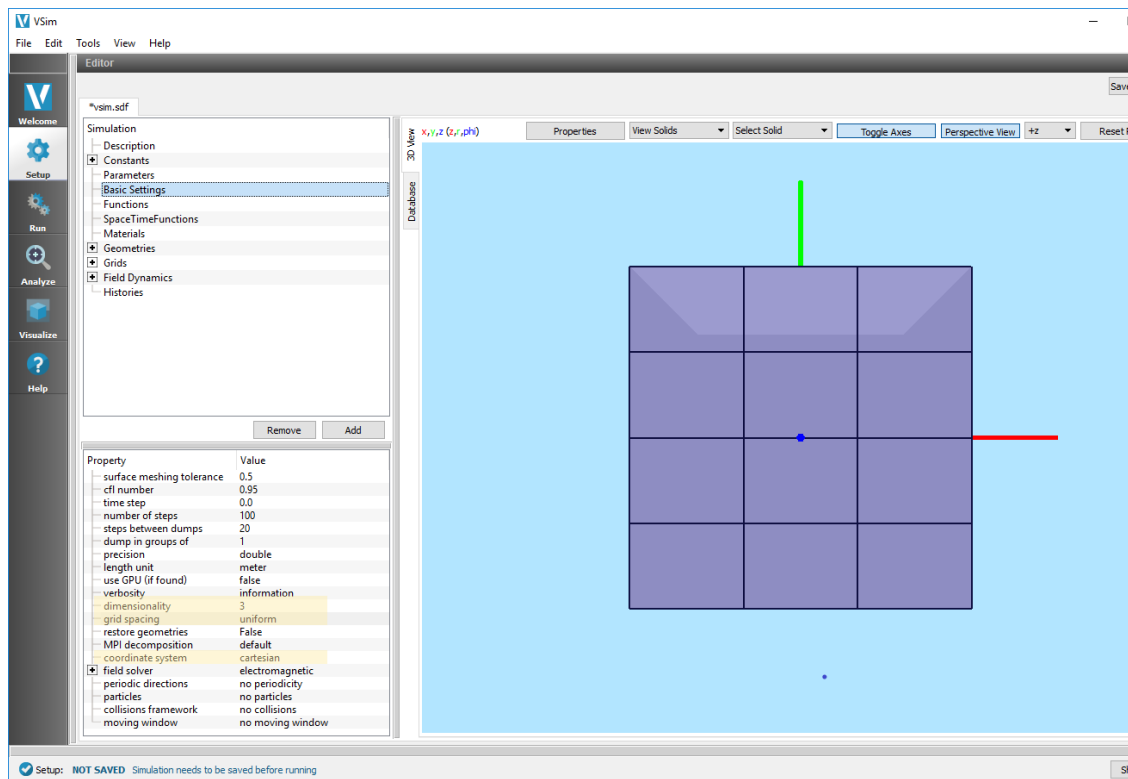


Fig. 8.17: Grid choices

The size of the domain can be set using the *Min* and *Max* properties of a uniform *Grid* element or with the minimum and maximum values of the *SectionBreaks* properties of a variable *Grid* element. The number of cells in each direction can also be specified. See Fig. 8.18.

---

**Note:** Only one grid may be added to any one simulation at a time.

---



---

**Note:** A grid can be resized to fit the bounds of a geometry by right clicking on the Grid element and choosing *Resize Grid*. Resizing the grid puts in numbers, replacing any constants and parameters.

---

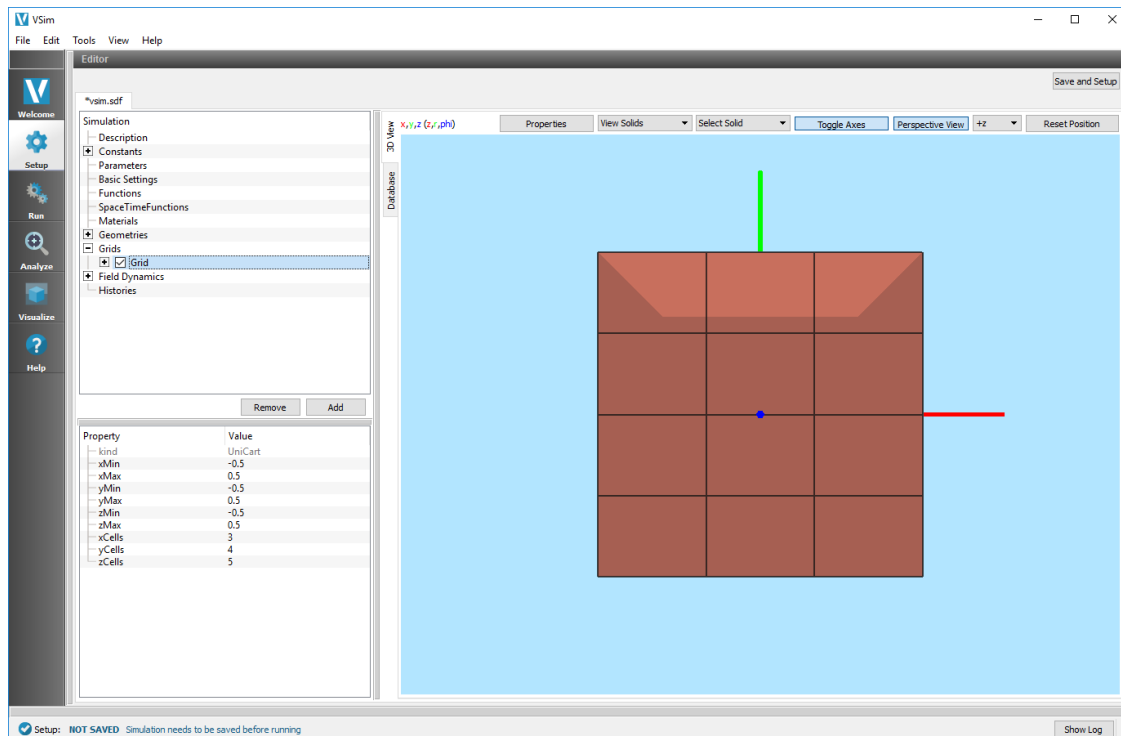


Fig. 8.18: Grid settings

## 8.13 Field Dynamics

The type of *field solver* is determined in the *Basic Settings* element. Its parameters are specified in the *Field Dynamics* element.

### 8.13.1 Field Solver

4 field solvers are available in Vsim

#### Electromagnetic Solver

The electromagnetic solver will solve the E, B and if explicitly included (or particles are in the simulation) a J field is added to the simulation.

#### Electrostatic Solver

The electrostatic solver will solve an E, Phi and Charge Density, with the option to include external magnetic fields.

### Prescribed Fields Solver

The prescribed field solver can be used to import previously solved external fields, and then provide a time dependent variation. This can be used when studying particle dynamics, particularly in multipacting simulations.

### Constant Fields Solver

The constant field solver can be used to apply a constant field in particle simulations. Particle movement will not impact these fields, however the field will impact particle movement.

### No Field Solver

A no field solver is used if only Particle Dynamics are to be studied, without the impact of fields.

### Hybrid Fluid Solver

The hybrid fluid solver enables hybrid fluids, which cannot interact with standard field solves.

## 8.13.2 Fields

Depending on the type of solver chosen, default fields will be initialized in the simulation. For an electrostatic simulation, default fields will be *Phi*, *Charge Density* and *Electric Field*. You can optionally add a *Background Charge Density* field or *External Field* by clicking the *Add → Add Field* button located at the bottom of the *Elements Tree*, or simply right clicking on the *Fields* element → *Add Field*.

An *External Field* is used for importing a *Magnetic* field in electrostatic simulations with particles.

For an electromagnetic simulation, default fields will be *Electric Field* and *Magnetic Field*. You can optionally add a *Current Density* field or *External Field* by clicking the *Add → Add Field* button located at the bottom of the *Elements Tree*, or simply right clicking on the *Fields* element → *Add Field*.

An *External Field* in electromagnetic simulations can be a *Magnetic*, *Electric*, or *Current* field. External fields are used to effect particle movements in simulations.

### Field Initial Conditions

An initial condition can be added to any field by clicking the *Add → Add FieldInitialCondition* button located at the bottom of the *Elements Tree*, or simply right clicking on the particular *Field* element → *Add FieldInitialCondition*. See Fig. 8.19.

### Field Boundary Conditions

A boundary condition can be added to any field by clicking the *Add → Add FieldBoundaryCondition* button located at the bottom of the *Elements Tree*, or simply right clicking on the particular *Field* element → *Add FieldBoundaryCondition*.

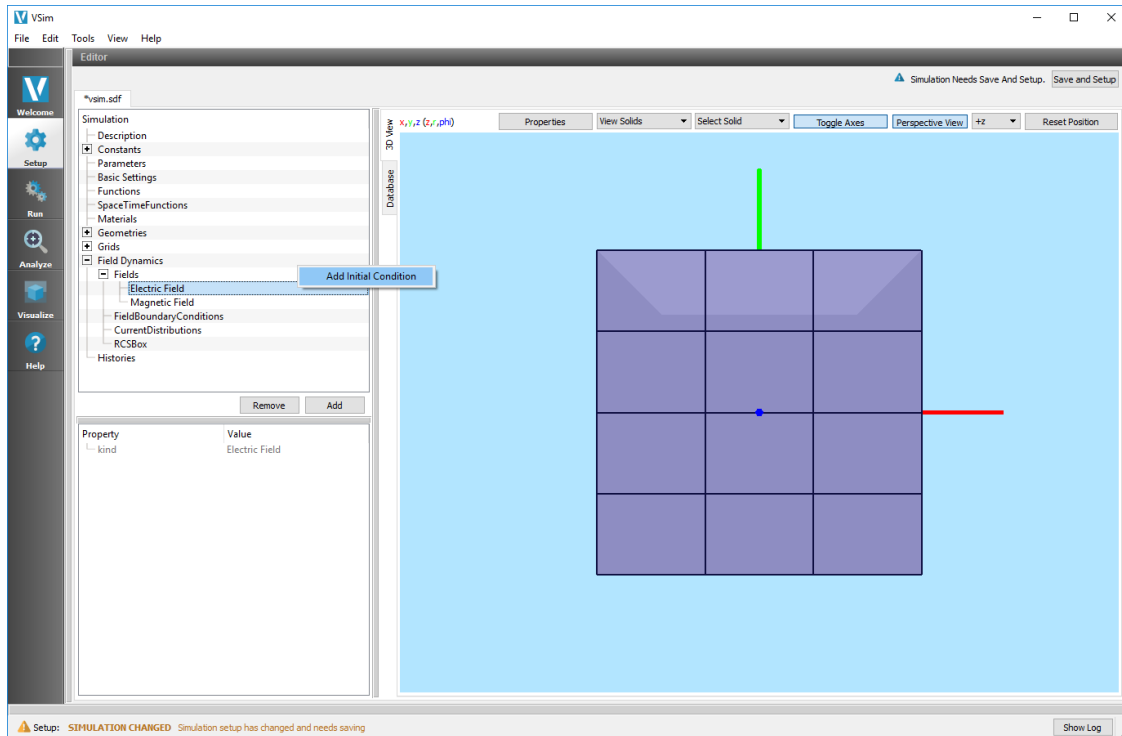


Fig. 8.19: Adding an initial condition to a field

## Current Distributions

A current distribution can be added to any field by clicking the *Add* → *Add CurrentDistribution* button located at the bottom of the *Elements Tree*, or simply right clicking on the particular *Field* element → *Add CurrentDistribution*.

A *distributedCurrent* is a volume current source where you can provide the min and max values in each direction. A *distributedCurrent* will show up on the *Geometry View*. You can hide a *distributedCurrent* by unchecking the box next to it. Hiding a current will not remove it from the simulation, it'll just hide it from the geometry view. See Fig. 8.20.

## Poisson Solver

The type of Poisson solve and any preconditioner can be set under the *solver* and *preconditioner* properties of the *Properties Editor*.

---

**Note:** Changing the solver type may introduce more properties due to the context-sensitive nature of the input.

---

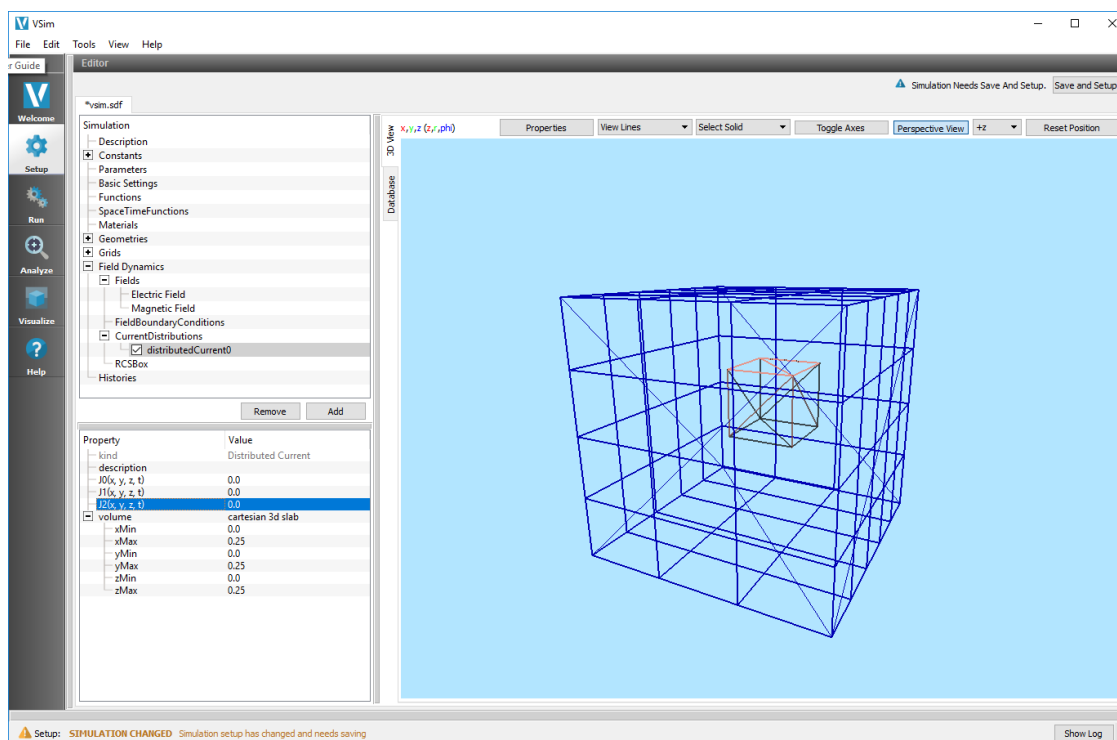


Fig. 8.20: A volume of current is shown in the smaller/interior box. The outer box is the grid.

## Reactions

To include Reactions in the Visual Setup, first select the *Basic Settings* element of the setup tree and ensure that the *particles* dropdown menu is set to “include particles” and the *collisions framework* dropdown is set to “reactions.”

With these settings selected, collisions can now be set up within the *Particle Dynamics* element of the setup tree. The Reactions are organized between five options: Particle Particle Collisions, Particle Fluid Collisions, Three Body Reactions, Field Ionization Processes, and Decay Processes. By highlighting one of these five options, right clicking, and adding a collision process a user is setting a `RxnProductGenerator`` (see :ref:`Vsim Reference Manual: Text Setup: Reactions <rxn-rxnProductGenerator>` for more information on `RxnProductGenerators`).

**Note:** The distinction between the Particle Particle Collisions and Particle Fluid Collisions is artificial in the visual setup and is made for the convenience of the user.

After adding a collision process to the tree, the user then selects the reacting species, cross-sections/reaction rates, and other reaction attributes. The species and fluids in the drop down menu for reactants and products are limited such that only selections appropriate for the process are available. Charge and mass conservation is checked during the translation from .sdf to .in. If there is a charge or mass violation, an error will be thrown.

If the drop-down menu used to set the interacting particle species is empty, make sure you’ve added the necessary `KineticParticle` or `BackgroundGas` for the type of collision.

## Reduced Collisions (Impact Collider)

To include Reduced Collisions in the Visual Setup, first select the *Basic Settings* element of the setup tree and ensure that the *particles* dropdown menu is set to “include particles” and the *collisions framework* dropdown is set to “reduced”. With these settings selected, collisions can now be set up within the *Particle Dynamics* element of the setup tree.

To add collisions between kinetic particles and a neutral background gas (fluid), click the *Add → Add ParticleFluidCollision* button located at the bottom of the *Elements Tree*, or simply right click on the *Collisions* element and select *Add ParticleFluidCollision* and choose whether either *Electron Neutral Fluid Collision* or *Ion Neutral Fluid Collision*. After making a selection, a new element will appear in the tree. Choose the particle species and the background gas that will interact.

To add a specific collision process, highlight this new element and choose a specific collision process from the *Add CollisionProcess* menu which will appear next to the mouse arrow. When a specific collision process is added, a new element will appear. The cross-sections for the interaction process will be set in this element.

See Fig. 8.21 for an example of adding collisions to a simulation in the Visual Setup.

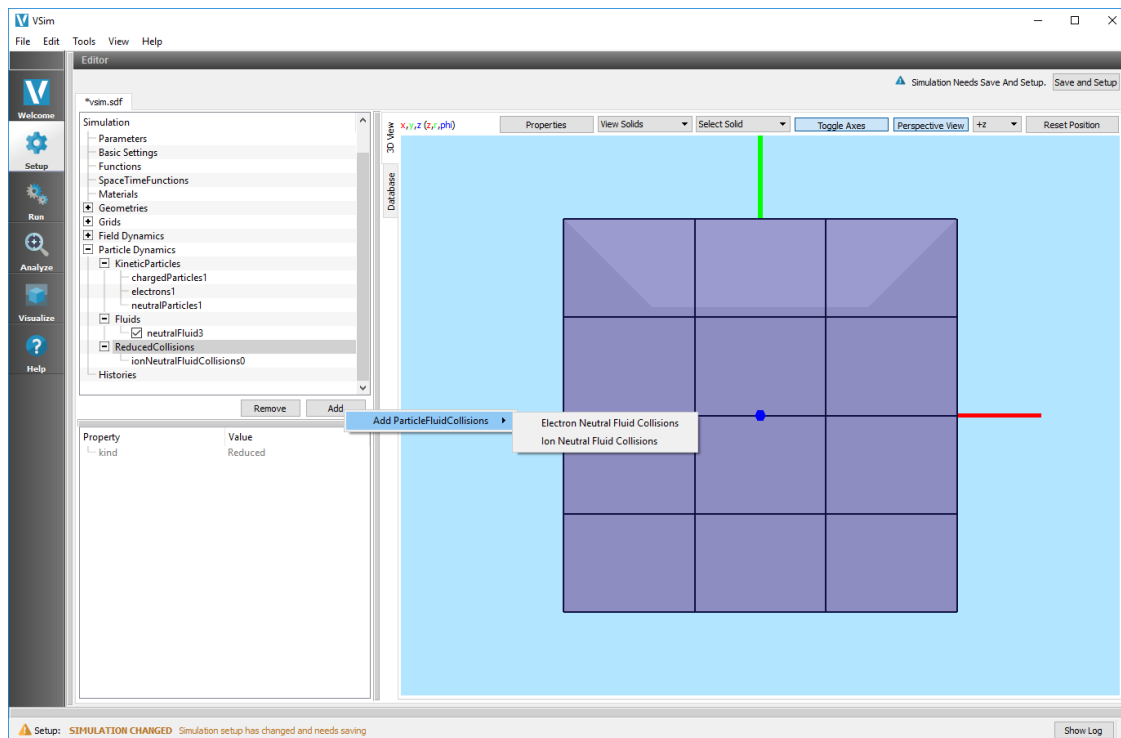


Fig. 8.21: Collisions

## Monte Carlo Interactions

To include Monte Carlo Interactions in the Visual Setup, first select the *Basic Settings* element of the setup tree and ensure that the *particles* dropdown menu is set to “include particles” and the *collisions framework* dropdown is set to “monte carlo”.

With these settings selected, collisions can now be set up within the *Particle Dynamics* element of the setup tree. The Monte Carlo are organized between five options: Particle Particle Collisions, Particle Fluid Collisions, Three Body Reactions, Field Ionization Processes, and Decay Processes. The user can add a specific interaction process by highlighting one of these five options, right clicking, and selecting a process from the *Add CollisionType* menu.

After adding a collision process to the tree, the user then selects the reacting species, cross-sections/reaction rates, and other reaction attributes.

## 8.14 Histories

Histories are used to calculate and record data about fields and particles in a simulation.

### 8.14.1 FieldHistory

*Field Histories* record on a per time-step basis. Field histories are used to measure quantities such as the value or energy of the field at a location. The output will be a 1D array of the value vs time.

Possible *Field Histories* include:

### 8.14.2 ArrayHistory

An *Array History* will output an array of data for each time-step.

Possible *Array Histories* include:

- Far-Field Box Data
- Field Slab Data

## 8.15 Property Editor

The *Property Editor* allows for setting of specific properties under each of the *Elements* from the *Elements Tree* for the simulation. Such properties might include sizes in the X, Y, and Z directions, the type of particles, and specifics of a field solve.

The *Property* and *Value* inputs are context-sensitive. The availability of a particular property may depend on other properties or selections of the *Element Tree*. For instance, changing the solver value in the *PoissonSolver* element will bring up a new set of solver *properties* to be set. Refer to the figure *Constructive Solid Geometry (CSG) Boolean Operator* (Fig. 8.12) to see the same principle with regard to geometry options.

## 8.16 Geometry View

The *3D View* section can be used to view the simulation setup including the geometry, grid, and source.

### 8.16.1 Buttons

- Properties

This will open a dialog that will allow editing the properties of the *3D View*. Following changes can be made through this dialog: Set line width and point size. Choose how Colors will be assigned to shapes in the *3D View*. See Fig. 8.9 for further information Set shininess of a shapes. Show or hide axes Show the coordinates of a clicked point of a geometry.



- View Solids

This will show all objects as solid, can also be set to View Solids/Lines, View Lines (wireframe) or View Points

- Distance This will allow measuring distance between two points of a geometry. The clicked points and the distance will be visible at the bottom of the *3D View*.

- Clip

This will open up a drop down menu to clip the simulation view, with a chosen normal direction and coordinate to place the clip at.

- Toggle Axes

A toggle button to show or hide the axes.

- Show Scale

This will allow changing the scale shown by default, from nanometers to kilometers. Note that all values will remain in meters or as defined by the user, this is just used to rapidly change the scale shown in the setup menu.

- Perspective View

This will reset to a perspective view for the 3D view.

- Axis Drop-down Menu

Drop-down menu for choosing the axis from which the object is viewed.

- Reset Position

Pressing this button will reset the camera view to be along the axis selected in the drop-down menu.

- Edit Mode

Edit mode will allow application of transformations to a selected portion of a geometry. Here is the workflow: After switching to edit mode, click on “Add Volume Selector” to add a selector, then move the selector to desired location to overlap the geometry by changing its position through the property editor. After that, select the geometry node that is to be edited and click on “Perform Selection”. A new node will be added in the tree with the selected part being red in color. All the transformations that are applied to this new node only gets applied to the red portion of the geometry. Click on “Done Editing” to exit this mode.

## 8.16.2 Navigating

Navigation in 3D space is possible under keyboard and mouse control.

- Rotate

Holding down the right mouse button and dragging it will rotate the camera position around the object being viewed.

- Pan

Holding down the left mouse button and the Shift key pans the view in a plane without rotating the viewed object.

- Zoom

Using the mouse wheel, the view can be zoomed in or out. If the system does not have a mouse wheel, then holding down the left mouse button and the Control key and moving the mouse up and down will also zoom in and out.

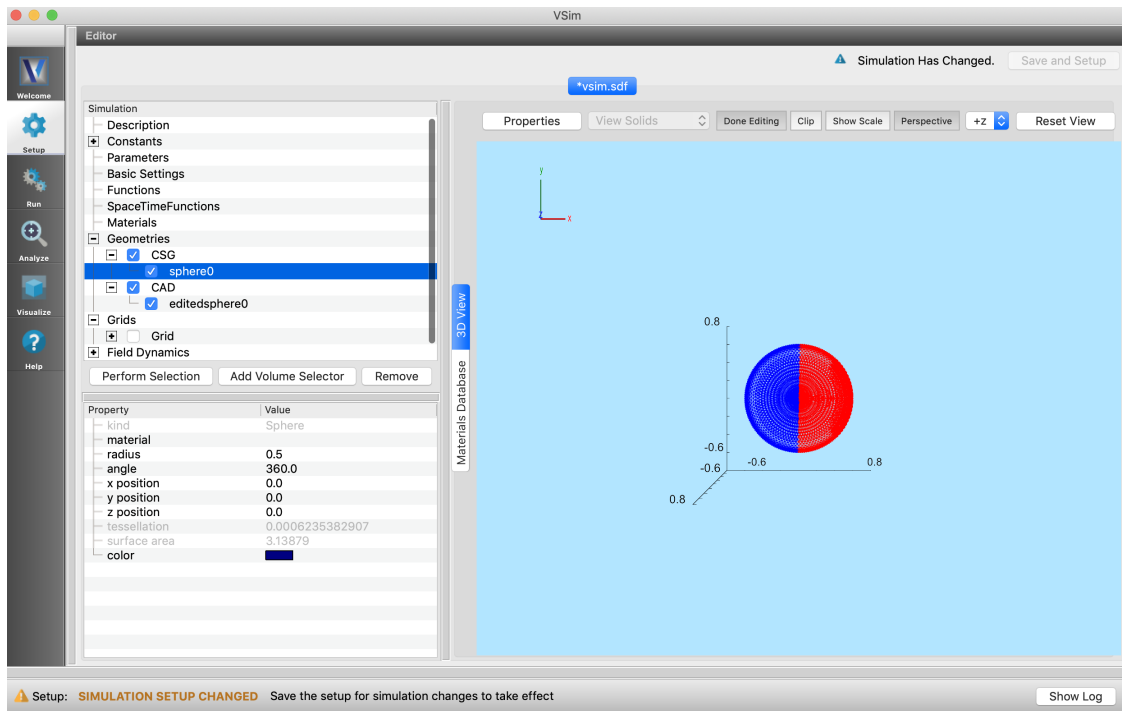


Fig. 8.22: The Partially Selected Geometry.

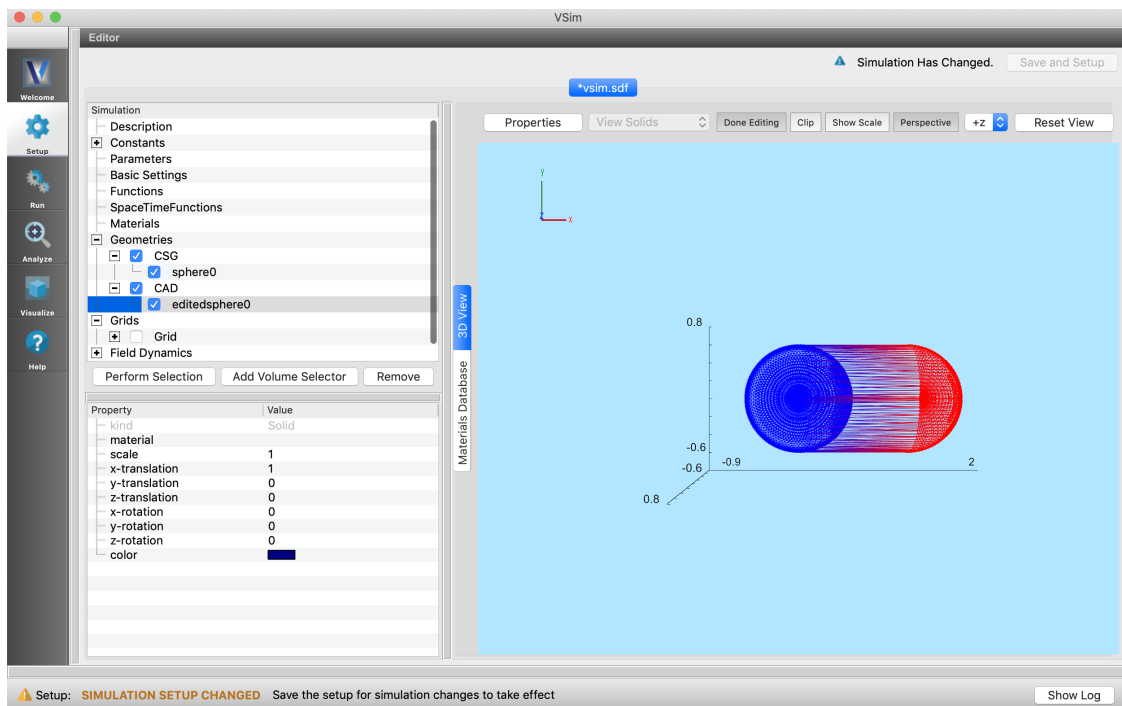


Fig. 8.23: The Partially Transformed Geometry.

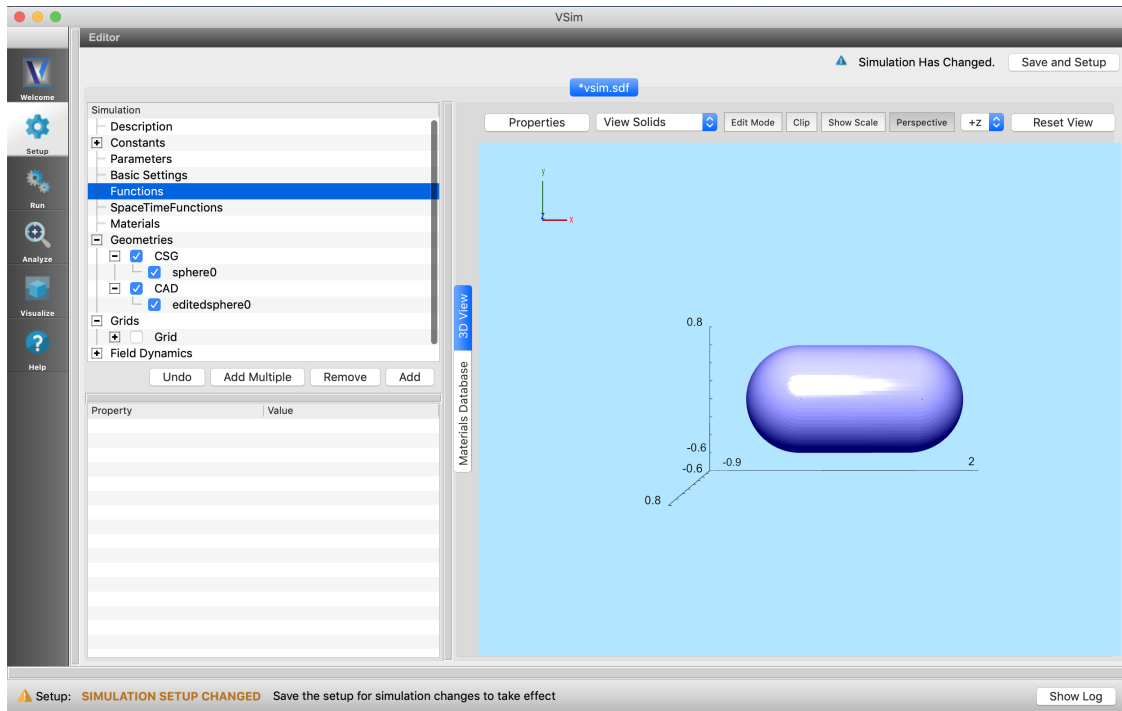


Fig. 8.24: The Final Geometry.

### 8.16.3 Database View

The *Database View* is for viewing and adding materials to your simulation. Initially, the *Database View* is blank, but upon importing a materials (.vmat) file, the view is populated with a table of data from the file. See Fig. 8.25.

You can switch between files, if more than one file is open, by changing the left drop-down menu. You can remove a file by first switching to the file you would like to close, and then clicking on the *Remove File* button.

You can add materials to your simulation by highlighting the particular material you are interested in, and then clicking on the *Add to Simulation* button.

For more information on materials, please see [Materials](#).

## 8.17 Particle Dynamics

The inclusion of *Particle Dynamics* is determined by the value of *particles* in the *Basic Settings* element. If *particles* is set to *no particles*, then no particles are in the simulation and the *Particle Dynamics* element is hidden.

If *particles* is set to *include particles* then the *Particle Dynamics* element is shown and further properties can be set.

The *Particle Dynamics* element holds information on any kinetic particles, background gases, and collisions in the simulation.

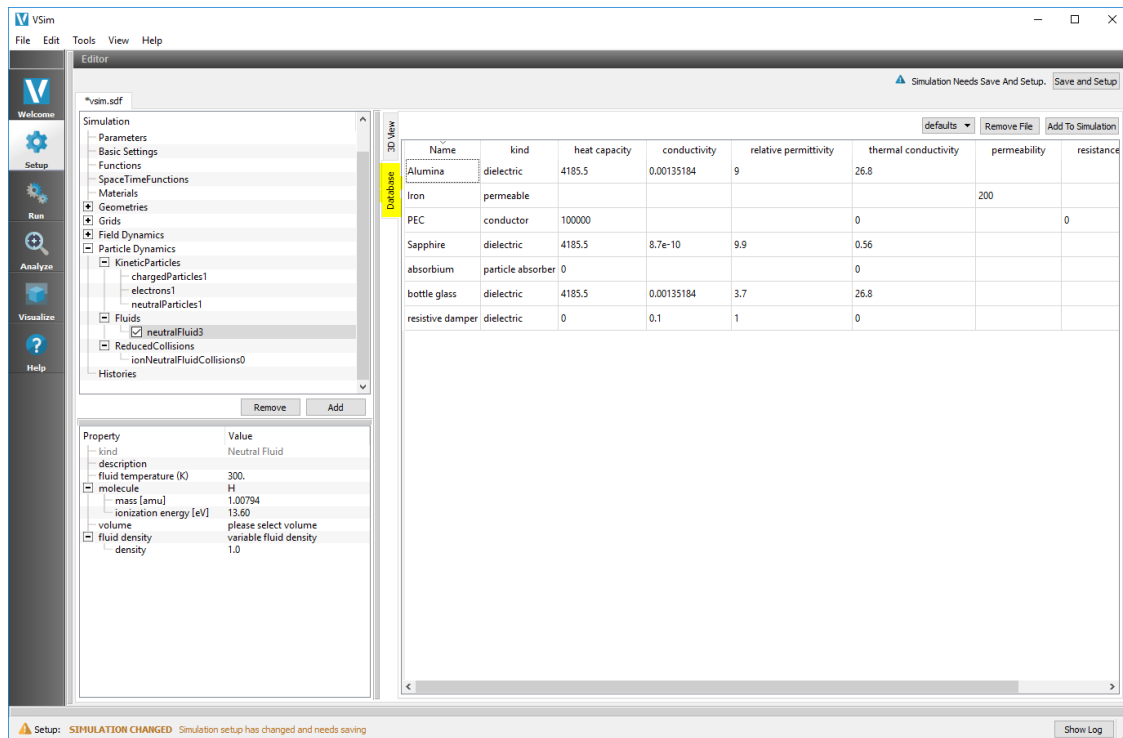


Fig. 8.25: The Database view

### 8.17.1 Kinetic Particles

Electrons, charged particles, and neutral particles can be added to the *KineticParticles* element.

To add kinetic particles, click the *Add → KineticParticle* button located at the bottom of the *Elements Tree*, or simply right click on the *KineticParticles* element and select *Add KineticParticle* and then choose the type of particle you want to include.

The properties of each kind of *KineticParticle* are modifiable in the *Properties Editor* pane.

#### Charged Particles

Charged Particles can be used to define any kinetic particle with given mass and charge.

To add Charged Particles, right click on the “*KineticParticles*” element, hover over the “*Add ParticleSpecies*” and choose “*Charged Particles*”.

##### kind (not editable)

Charged Particles

##### nominal density

A positive value suggesting the nominal density for the particles. This will be used in conjunction with the weight setting to compute the density, weights, and number of particles in a macro particle for your kinetic particles.

##### description

A space to provide a descriptive comment for the particle species.

##### particle dynamics

Whether to use relativistic or non-relativistic particles.

- **relativistic:** Use the relativistic particle pushing algorithm to update the particle positions and velocities by including a gamma term.
- **non-relativistic:** Use the non-relativistic particle pushing algorithm to update the particle positions and velocities.

### particle weights

Whether to use constant or variable weight particles.

- **variable weights:** The weights of the macroparticles can vary throughout the simulation.
- **constant weights:** The weights of the macroparticles are constant throughout the simulation.
- **managed weights:** Variable weight particles that are managed to allow for maximum and minimum weights, and maximum and minimum number of macroparticles per cell. Note if working in cylindrical coordinates, some computational artifacts may arise in a few simulation types with managed weights. Contact Tech-X support if this becomes a concern.

#### macroparticles per cell for splitting

If more than this many macroparticles are in a cell splitting will not occur. This is converted into a STFunc which is set to a constant in Cartesian coordinates. In cylindrical coordinates, it is important to recognize that the volume of a cell is a function of radius. Therefore, when a gas is in equilibrium, then statistically one expects to find fewer particles in a cell for small values of R. As a result, to maintain an equilibrium density in cylindrical coordinates, this thresholdFunc would be smaller for small values of R and larger for large values of R. The following function is used to determine the number of particles per cell for splitting:  $\max(1.0, THRESHOLD * 10^{1.5 * (y/MAXR) - 0.75})$ , where THRESHOLD is the value you specify in Composer, and MAXR is the maximum R grid value. If you want flexibility in adjusting this function, please contact Tech-X customer support.

#### macroparticles per cell for combining

If fewer than this many macroparticles are in a cell combining will not occur. All other aspects to this capability are otherwise similar to the “macroparticles per cell for splitting” described above.

#### minimum split particle weight

If the split particles would weigh under this value, splitting will not occur.

#### maximum combined particle weight

If the combined particle would weigh over this value, combination will not occur.

#### splitting periodicity

Number of time steps between assessing if particles should be split.

#### combining periodicity (not editable)

Number of time steps between assessing if particles should be combined.

#### splitting algorithm

Algorithm to use in determining split particle weights. This interaction is designed to use variable-weight species. Kind of MonteCarlo interaction that splits macroparticles of a single species into two or more macroparticles. This interaction is applied at each time step. The attribute `algorithm_kind` specifies the type of algorithm to be used when splitting macroparticles. The possible values are 1 and 2. Algorithm 1 (default) performs a simple, fast split operation, whereas option 2 is more accurate but more computationally intensive. In algorithm 1, the weight of the macro particle is simply split to one half and a new macroparticle gets added at the same location with one half weight value. In algorithm 2, the split operation is done such that the number of macro particles within a cell meets the requirement of the threshold value specified by the user.

**combining algorithm**

Algorithm to use in determining combined particle weights. Kind of MonteCarlo interaction that combines macroparticles of a single species together into one single macroparticle. The `algorithm_kind` specifies the kind of algorithm to be used when combining macroparticles. The possible values are 1, 2, 3, 4, 5, or 6. Option 1 indicates a simple pair-wise combination of macroparticles, deleting one and adding the number of real particles (weight) to the other (decimation). Option 2 indicates a slightly more accurate pair-wise approach, deleting one macroparticle and assigning the other the mean position and velocity of the pair as well as the total weight of the pair (inelastic). Option 3 is an elastic combination approach, where quartets of macroparticles are combined into pairs that conserve energy and momentum (elastic). Option 4 is a pair-wise combining approach similar to Option 2 except that it allows users to avoid combining particles that are above the user specified limit. Option 5 is a pairwise approach in which the particles within a cell are grouped in velocity phase-space bins and then particles in each velocity phase-space bin are combined pair-wise. In this approach energy and momentum are conserved. Option 6 is a fluxConserving approach, combining 3+ particles into 2. This has enough degrees of freedom to conserve energy, momentum and a third quantity. We chose the center of momentum frame flux, which has the nice property that the solution is always real. This conserves the original phase space of the particles.

**maintain equilibrium (cylindrical only)**

If set to True, in cylindrical coordinates the combining and splitting thresholds will be modified to the following equation.  $\max(1.0, \text{threshold} * 10^{*(1.5 * y / \text{LY} - 0.75)})$  This attempts to handle the fact that cylindrical cells will increase in size as the value of `r` increases.

---

**Note:** This feature replaces the `selfCombCollision` previously implemented in Vorpal collision.

---

**weight setting**

Whether to use computed weights or explicitly set weights.

- **computed weights:**

Let VSim calculate your macroparticle weights for you based on the number of macroparticles per cell you specify as well as the nominal density. The weights are calculated such that the number of particles in a macro particle is equal to the  $\text{nominalDensity} * \text{cellVolume} / \text{macroparticles per cell}$ .

**macroparticles per cell:** The number of macroparticles per cell.

- **explicitly set weights:**

Declare the number of particles in a macro particle explicitly.

**particles per macroparticle:** The number of particles in a macroparticle.

**store species current density**

In cartesian coordinate electrostatic simulations, this can be set to “store current density”. This will create a field for storing the particle currents which can be a useful diagnostic.

**molecule**

Molecule of the charged particle. A custom ion is available for those not pre-defined.

- **mass [amu]**

The mass of a single real particle in atomic mass units.

- **charge number**

The charge number of a single particle, multiple of the fundamental charge.

- **ionization energy [eV]**

The ionization energy of the molecule in electron volts. This value will be used by particle interactions set thresholds and determine energy losses.

**Additional Features:**

Emitters:

- *Settable Shape Flux Emitter*
- *Settable Slab Flux Emitter*

Boundary Conditions:

- *Properties of Particle Boundary Conditions*

Loaders:

- *Particle Loader*

**Electrons**

Electrons are pre-defined to have a charge =  $-1.602176487 \times 10^{-19}$  C and a mass =  $9.10938215 \times 10^{-31}$  kg.

To add Electrons, right click on the “KineticParticles” element, hover over the “Add ParticleSpecies” and choose “Electrons”.

**kind (not editable):**

Electrons

**nominal density:**

A positive value suggesting the nominal density for the particles. This will be used in conjunction with the weight setting to compute the density, weights, and number of particles in a macro particle for your kinetic particles.

**description**

A space to provide a descriptive comment for the particle species.

**particle dynamics:**

Whether to use relativistic or non-relativistic particles.

- **relativistic:** Use the relativistic particle pushing algorithm to update the particle positions and velocities by including a gamma term.
- **non-relativistic:** Use the non-relativistic particle pushing algorithm to update the particle positions and velocities.
- **speed limited:** Only available in constant field cartesian simulation with variable weights, speed limited particles are a variation of non-relativistic and weights will evolve differently during a simulation

**speed limit:**

Default is lightspeed, this is the maximum speed particles can go.

**integration tolerance**

The particle integration tolerance.

**speed limiting function**

At this time the only option available is abrupt.

**particle weights:**

Whether to use constant or variable weight particles.

- **variable weights:** The weights of the macroparticles can vary throughout the simulation.
- **constant weights:** The weights of the macroparticles are constant throughout the simulation.

- **managed weights:** Variable weight particles that are managed to allow for maximum and minimum weights, and maximum and minimum number of macroparticles per cell. Note if working in cylindrical coordinates, some computational artifacts may arise in a few simulation types with managed weights. Contact Tech-X support if this becomes a concern.

**macroparticles per cell for splitting:**

If more than this many macroparticles are in a cell splitting will not occur. This is converted into a STFunc which is set to a constant in Cartesian coordinates. In cylindrical coordinates, it is important to recognize that the volume of a cell is a function of radius. Therefore, when a gas is in equilibrium, then statistically one expects to find fewer particles in a cell for small values of R. As a result, to maintain an equilibrium density in cylindrical coordinates, this thresholdFunc would be smaller for small values of R and larger for large values of R. The following function is used to determine the number of particles per cell for splitting:  $\max(1.0, THRESHOLD * 10^{(1.5 * (y/MAXR) - 0.75)})$ , where THRESHOLD is the value you specify in Composer, and MAXR is the maximum R grid value. If you want flexibility in adjusting this function, please contact Tech-X customer support.

**macroparticles per cell for combining:**

If fewer than this many macroparticles are in a cell combining will not occur. All other aspects to this capability are otherwise similar to the “macroparticles per cell for splitting” described above.

**minimum split particle weight:**

If the split particles would weigh under this value, splitting will not occur.

**maximum combined particle weight:**

If the combined particle would weigh over this value, combination will not occur.

**splitting periodicity:**

Number of time steps between assessing if particles should be split.

**combining periodicity (not editable):**

Number of time steps between assessing if particles should be combined.

**splitting algorithm:**

Algorithm to use in determining split particle weights.

**combining algorithm:**

Algorithm to use in determining combined particle weights.

**maintain equilibrium (cylindrical only)**

If set to True, in cylindrical coordinates the combining and splitting thresholds will be modified to the following equation.  $\max(1.0, \text{threshold} * 10^{(1.5 * y / LY - 0.75)})$  This attempts to handle the fact that cylindrical cells will increase in size as the value of r increases.

**weight setting:**

Whether to use computed weights or explicitly set weights.

- **computed weights:**

Let VSim calculate your macroparticle weights for you based on the number of macroparticles per cell you specify as well as the nominal density. The weights are calculated such that the number of particles in a macro particle is equal to the nominalDensity \* cellVolume / macroparticles per cell.

**macroparticles per cell:** The number of macroparticles per cell.

- **explicitly set weights:**

Declare the number of particles in a macro particle explicitly.

**particles per macroparticle:** The number of particles in a macroparticle.



**store species current density**

In cartesian coordinate electrostatic simulations, this can be set to “store current density”. This will create a field for storing the particle currents which can be a useful diagnostic.

**Additional Features:**

Emitters:

- *Settable Shape Flux Emitter*
- *Secondary Electron Emitter*
- *Settable Slab Flux Emitter*

Boundary Conditions:

- *Properties of Particle Boundary Conditions*

Loaders:

- *Particle Loader*

**Neutral Particles**

Neutral Particles of the *gas kinds* listed below are pre-defined with the appropriate charge and mass values. Users can also add custom neutral species.

To add Neutral Particles, right click on the “KineticParticles” element, hover over the “Add ParticleSpecies” and choose “Neutral Particles”.

**kind (not editable)**

Neutral Particles

**nominal density**

A positive value suggesting the nominal density for the particles. This will be used, in conjunction with the weight setting, to compute the density, weights, and number of particles in a macro particle for your kinetic particles.

**description**

A space to provide a descriptive comment for the particle species.

**particle dynamics**

Whether to use relativistic or non-relativistic particles

- **relativistic:** Use the relativistic particle pushing algorithm to update the particle positions and velocities by including a gamma term.
- **non-relativistic:** Use the non-relativistic particle pushing algorithm to update the particle positions and velocities.

**particle weights**

Whether to use constant or variable weight particles.

- **variable weights**  
The weights of the macroparticles can vary throughout the simulation.
- **constant weights**  
The weights of the macroparticles are constant throughout the simulation.

- **managed weights**

Variable weight particles that are managed to allow for maximum and minimum weights, and maximum and minimum number of macroparticles per cell. Note if working in cylindrical coordinates, some computational artifacts may arise in a few simulation types with managed weights. Contact Tech-X support if this becomes a concern.

**macroparticles per cell for splitting**

If more than this many macroparticles are in a cell splitting will not occur. This is converted into a STFunc which is set to a constant in Cartesian coordinates. In cylindrical coordinates, it is important to recognize that the volume of a cell is a function of radius. Therefore, when a gas is in equilibrium, then statistically one expects to find fewer particles in a cell for small values of R. As a result, to maintain an equilibrium density in cylindrical coordinates, this thresholdFunc would be smaller for small values of R and larger for large values of R. The following function is used to determine the number of particles per cell for splitting:  $\max(1.0, THRESHOLD * 10^{(1.5 * (y/MAXR) - 0.75)})$ , where THRESHOLD is the value you specify in Composer, and MAXR is the maximum R grid value. If you want flexibility in adjusting this function, please contact Tech-X customer support.

**macroparticles per cell for combining**

If fewer than this many macroparticles are in a cell combining will not occur. All other aspects to this capability are otherwise similar to the “macroparticles per cell for splitting” described above.

**minimum split particle weight**

If the split particles would weigh under this value, splitting will not occur.

**maximum combined particle weight**

If the combined particle would weigh over this value, combination will not occur.

**splitting periodicity**

Number of time steps between assessing if particles should be split.

**combining periodicity (not editable)**

Number of time steps between assessing if particles should be combined.

**splitting algorithm**

Algorithm to use in determining split particle weights.

**combining algorithm**

Algorithm to use in determining combined particle weights.

**maintain equilibrium (cylindrical only)**

If set to True, in cylindrical coordinates the combining and splitting thresholds will be modified to the following equation.  $\max(1.0, \text{threshold} * 10^{(1.5 * y/LY - 0.75)})$  This attempts to handle the fact that cylindrical cells will increase in size as the value of r increases.

**weight setting**

Whether to use computed weights or explicitly set weights.

- **computed weights**

Let VSim calculate your macroparticle weights for you, based on the number of macroparticles per cell you specify as well as the nominal density. The weights are calculated such that the number of particles in a macro particle is equal to the nominalDensity \* cellVolume / macroparticles per cell.

**macroparticles per cell:** The number of macroparticles per cell.

- **explicitly set weights**

Declare the number of particles in a macro particle explicitly.

**particles per macroparticle:** The number of particles in a macroparticle.

**molecule**

Molecule of the neutral particle. Options are:

- **H**: Hydrogen (atomic)
- **H2**: Hydrogen (molecular)
- **He**: Helium
- **Ar**: Argon
- **Xe**: Xenon
- **Rn**: Radon
- **Kr**: Krypton
- **O**: Oxygen (atomic)
- **O2**: Oxygen (molecular)
- **Ne**: Neon
- **N**: Nitrogen (atomic)
- **N2**: Nitrogen (molecular)
- **custom molecule**
  - Set the mass (in amu) and ionization energy (in eV) of a custom gas.
  - **mass [amu]** The mass of a single real particle in atomic mass units.
  - **ionization energy [eV]** The ionization energy of the molecule in electron volts.

**Additional Features:**

Emitters:

- *Settable Shape Flux Emitter*
- *Sputter Emitter*
- *Settable Slab Flux Emitter*

Boundary Conditions:

- *Properties of Particle Boundary Conditions*

Loaders:

- *Particle Loader*

**Field Scaling Electrons**

These are a special type of electron used in testing. Most notably, they consist of only one physical particle per macroparticle. They are in fact given a variable weight, which is used to track whether the particles have been created via secondary emission. Each particle also has a scaling value, allowing for multiple voltage levels to exist in the simulation simultaneously. Since there is only one physical particle per macroparticle, these particles will have negligible effect on the simulation.

The primary use of these particles is to scan a structure for multipacting resonances across a number of power levels at the same time.

To add Field Scaling Electrons, right click on the “KineticParticles” element, hover over the “Add TestParticleSpecies” and choose “Field Scaling Electrons”.

### Field Scaling Electron Loader

This particle loader is very similar to a standard particle species loader, with some notable exceptions.

**description:** A space to provide a descriptive comment for the particle species.

**load density:** Only the relative density of particles may be specified, as this is the same as the physical density.

**physical density:** This will specify the number of particles per cell.

**particle load placement:** At this time particles may only be loaded according to a bit-reversed algorithm.

**load duration:** Whether to only load the particles at the beginning of the simulation or repeatedly.

- **initialize only:** Particles will only be loaded at the beginning of the simulation.
- **repeat loading:** Particles will be loaded according to the parameters below.

**start time:** The time at which to start loading particles.

**stop time:** The time at which to stop loading particles.

**load after initialization:** Loading will repeat in all cells during the loading period.

**load upon shift:** Load particles into cells brought into the simulation by a moving window.

**scaling factor:** This will determine the minimum value, maximum value, and scaling factor applied to the field scaling electrons. So for example with a minimum value of 10, and a scaling factor of 3, that individual electron is charged to 30V.

**minimum scaling factor:** The minimum voltage of the particles loaded.

**maximum scaling factor:** The maximum voltage of the particles loaded.

**number of scale factors:** The number of scale factors, or steps, between the minimum scaling factor and maximum scaling factor.

**volume:** The volume in which to load particles.

- **cartesian 3d slab**
- **cylindrical 2d slab**

### Additional Features:

Fewer additional features are available for Field Scaling Electrons. The only emitter available for Field Scaling Electrons is a secondary emitter.

- *Secondary Electron Emitter*

And the only available particle boundary conditions are: Absorbing, Boundary Absorb and Save, Cut-Cell Absorb and Save, Interior Absorb and Save, and Reflecting.

- *Properties of Particle Boundary Conditions*

## Particle Emitters

### Settable Shape Flux Emitter

All particle types may emit from a shape settable flux emitter. Certain emission specifications are only available based on particle type and particle weights specification. Available in cartesian coordinate simulations only.

**start time**

Time to start emitting particles.

**stop time**

Time to stop emitting particles.

**emission specification:** Specification of the emitted particles, note that the specification options vary for constant or variable/managed weight particles.

- **emission current density:**

Only available with variable or managed weight specified particles.

**emission current density:** Specify the current density of the emitter (amps/meter<sup>2</sup>). Can be a spatial profile.

**velocity coordinate system:** Either global or surface. A global coordinate system will specify the emission velocities according to global axis. A surface coordinate system will set the emission directions according to the normal of the emission object. So in a surface coordinate system a lower simulation bounds the emission velocity must be negative to emit into the simulation space, for an upper simulation boundary the particles must be positive to emit into the simulation space.

**mean velocity 0:** The average (mean) speed of particles in the x-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for the direction normal to the emitting surface.

**mean velocity 1:** The average (mean) speed of particles in the y-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for a direction perpendicular to the emitting surface.

**mean velocity 2:** The average (mean) speed of particles in the z-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for a direction perpendicular to the emitting surface.

**thermal velocity 0:** A spread (standard deviation) for particle speeds in the 0 direction.

**thermal velocity 1:** A spread (standard deviation) for particle speeds in the 1 direction.

**thermal velocity 2:** A spread (standard deviation) for particle speeds in the 2 direction.

- **emission flux:**

Only available with variable or managed weight specified particles.

**emission flux:** Specify the flux of the emitter (particles/meter<sup>2</sup>). Can be a spatial profile.

**velocity coordinate system:** Either global or surface. A global coordinate system will specify the emission velocities according to global axis. A surface coordinate system will set the emission directions according to the normal of the emission object. So in a surface coordinate system a lower simulation bounds the emission velocity must be negative to emit into the simulation space, for an upper simulation boundary the particles must be positive to emit into the simulation space.

**mean velocity 0:** The average (mean) speed of particles in the x-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for the direction normal to the emitting surface.

**mean velocity 1:** The average (mean) speed of particles in the y-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for a direction perpendicular to the emitting surface.

**mean velocity 2:** The average (mean) speed of particles in the z-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for a direction perpendicular to the emitting surface.

**thermal velocity 0:** A spread (standard deviation) for particle speeds in the 0 direction.

**thermal velocity 1:** A spread (standard deviation) for particle speeds in the 1 direction.

**thermal velocity 2:** A spread (standard deviation) for particle speeds in the 2 direction.

- **emission current:**

Only available with constant weight particles.

**emission current:** Specify the total emitted current per second from the emitter (amps/second).

**velocity coordinate system:** Either global or surface. A global coordinate system will specify the emission velocities according to global axis. A surface coordinate system will set the emission directions according to the normal of the emission object. So in a surface coordinate system a lower simulation bounds the emission velocity must be negative to emit into the simulation space, for an upper simulation boundary the particles must be positive to emit into the simulation space.

**mean velocity 0:** The average (mean) speed of particles in the x-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for the direction normal to the emitting surface.

**mean velocity 1:** The average (mean) speed of particles in the y-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for a direction perpendicular to the emitting surface.

**mean velocity 2:** The average (mean) speed of particles in the z-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for a direction perpendicular to the emitting surface.

**thermal velocity 0:** A spread (standard deviation) for particle speeds in the 0 direction.

**thermal velocity 1:** A spread (standard deviation) for particle speeds in the 1 direction.

**thermal velocity 2:** A spread (standard deviation) for particle speeds in the 2 direction.

- **emission rate:**

Only available with constant weight particles.

**emission rate:** Specify the total emitted particles per second from the emitter (particle/second).

**velocity coordinate system:** Either global or surface. A global coordinate system will specify the emission velocities according to global axis. A surface coordinate system will set the emission directions according to the normal of the emission object. So in a surface coordinate system a lower simulation bounds the emission velocity must be negative to emit into the simulation space, for an upper simulation boundary the particles must be positive to emit into the simulation space.

**mean velocity 0:** The average (mean) speed of particles in the x-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for the direction normal to the emitting surface.

**mean velocity 1:** The average (mean) speed of particles in the y-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for a direction perpendicular to the emitting surface.

**mean velocity 2:** The average (mean) speed of particles in the z-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for a direction perpendicular to the emitting surface.

**thermal velocity 0:** A spread (standard deviation) for particle speeds in the 0 direction.

**thermal velocity 1:** A spread (standard deviation) for particle speeds in the 1 direction.

**thermal velocity 2:** A spread (standard deviation) for particle speeds in the 2 direction.

- **Fowler Nordheim Emission:**

Specify particle emission according to the Fowler-Nordheim model (see the “Fowler-Nordheim Emitter” section in the Reference manual).. Only available with variable or managed weight electron particle species that are not speed-limited.

**work function [eV]:** Work function of the material from which emission is occurring.

**A:** Coefficient A of the Fowler-Nordheim emission model.

**B:** Coefficient B of the Fowler-Nordheim emission model.

**field enhancement:** Multiplies the measured electric field by this amount.

**Cv:** Coefficient Cv of the Fowler-Nordheim emission model.

**Cy:** Coefficient Cy of the Fowler-Nordheim emission model.

- **Richardson Dushman Emission:**

Specify particle emission according to the Richardson-Dushman model (see the “Richardson-Dushman Emitter” section in the Reference manual).. Only available with variable or managed weight electron particle species that are not speed-limited.

**work function [eV]:** Work function of the material from which emission is occurring. Parameter in the Richardson-Dushman model.

**field evaluation offset:** The offset from the surface where the field resulting from the particle is evaluated.

**temperature (K):** Temperature of the material from which emission is occurring. Parameter in the Richardson-Dushman model.

**field enhancement:** Multiplies the measured electric field by this amount.

**flux multiplier:** Multiplies the resulting output current by this amount.

- **Child Langmuir Emission:**

Specify particle emission according to the Child Langmuir model. Only available with variable or managed weight electron particle species that are not speed-limited.

**space charge limited emission:**

This will limit the current to provide a more consistent emission current, providing higher accuracy particularly in explosive emission cases, such as a pulsed power magnetron. For non pulsed-power simulations it is not necessary.

**average velocity 0:** The average (mean) speed of particles in the 0 direction.

**average velocity 1:** The average (mean) speed of particles in the 1 direction.

**average velocity 2:** The average (mean) speed of particles in the 2 direction.

**thermal velocity 0:** A spread (standard deviation) for particle speeds in the 0 direction.

**thermal velocity 1:** A spread (standard deviation) for particle speeds in the 1 direction.

**thermal velocity 2:** A spread (standard deviation) for particle speeds in the 2 direction.

### **emission surface**

The surface off of which to emit.

**object name:** The name of the geometry off of which to emit.

**emission offset:** The distance away from the object that emitted particles are placed, as a fraction of a cell length.

### **macroparticle emission:**

Only available with variable or managed weight particles. This allows for the specification of the macroparticle emission independent of the emitted particles. Used to handle computational concerns around macroparticle weight.

### **macroparticle rate:**

Number of macroparticles to emit per timestep. This value can be modified by the macroparticle emission profile.

### **macroparticle emission profile:**

Spatial profile for emission of the macroparticles. If this corresponded to half of the emissions shape, half of the number of macroparticles specified in *macroparticle rate* would be emitted, while the emission specification would be unaffected.

## **Settable Slab Flux Emitter**

All particle types may emit from a slab settable flux emitter. Certain emission specifications are only available based on particle type and particle weights specification. Available in all coordinate simulations.

**start time** Time to start emitting particles in seconds.

**stop time** Time to stop emitting particles in seconds.

### **emission specification: Specification of the emitted**

particles, note that the specification options vary for constant or variable/managed weight particles.

- **emission current density**

**emission current density:** Specify the current density of the emitter (amps/meter<sup>2</sup>). Can be a spatial profile.

**velocity coordinate system:** Either global or surface. A global coordinate system will specify the emission velocities according to global axis. A surface coordinate system will set the emission directions according to the normal of the emission object. So in a surface coordinate system a lower simulation bounds the emission velocity must be negative to emit into the simulation space, for an upper simulation boundary the particles must be positive to emit into the simulation space.

**mean velocity 0:** The average (mean) speed of particles in the x-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for the direction normal to the emitting surface.

**mean velocity 1:** The average (mean) speed of particles in the y-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for a direction perpendicular to the emitting surface.

**mean velocity 2:** The average (mean) speed of particles in the z-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for a direction perpendicular to the emitting surface.

**thermal velocity 0:** A spread (standard deviation) for particle speeds in the 0 direction.

**thermal velocity 1:** A spread (standard deviation) for particle speeds in the 1 direction.



**thermal velocity 2:** A spread (standard deviation) for particle speeds in the 2 direction.

- **emission flux**

**emission flux** Specify the flux of the emitter (particles/meter<sup>2</sup>). Can be a spatial profile.

**velocity coordinate system:** Either global or surface. A global coordinate system will specify the emission velocities according to global axis. A surface coordinate system will set the emission directions according to the normal of the emission object. So in a surface coordinate system a lower simulation bounds the emission velocity must be negative to emit into the simulation space, for an upper simulation boundary the particles must be positive to emit into the simulation space.

**mean velocity 0:** The average (mean) speed of particles in the x-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for the direction normal to the emitting surface.

**mean velocity 1:** The average (mean) speed of particles in the y-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for a direction perpendicular to the emitting surface.

**mean velocity 2:** The average (mean) speed of particles in the z-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for a direction perpendicular to the emitting surface.

**thermal velocity 0:** A spread (standard deviation) for particle speeds in the 0 direction.

**thermal velocity 1:** A spread (standard deviation) for particle speeds in the 1 direction.

**thermal velocity 2:** A spread (standard deviation) for particle speeds in the 2 direction.

- **emission current**

**emission current** Specify the total emitted current per second from the emitter (amps/second).

**profile** Spacetime function that multiplies the total emission current per second.

**velocity coordinate system:** Either global or surface. A global coordinate system will specify the emission velocities according to global axis. A surface coordinate system will set the emission directions according to the normal of the emission object. So in a surface coordinate system a lower simulation bounds the emission velocity must be negative to emit into the simulation space, for an upper simulation boundary the particles must be positive to emit into the simulation space.

**mean velocity 0:** The average (mean) speed of particles in the x-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for the direction normal to the emitting surface.

**mean velocity 1:** The average (mean) speed of particles in the y-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for a direction perpendicular to the emitting surface.

**mean velocity 2:** The average (mean) speed of particles in the z-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for a direction perpendicular to the emitting surface.

**thermal velocity 0:** A spread (standard deviation) for particle speeds in the 0 direction.

**thermal velocity 1:** A spread (standard deviation) for particle speeds in the 1 direction.

**thermal velocity 2:** A spread (standard deviation) for particle speeds in the 2 direction.

- **emission rate**

**emission rate** Specify the total number of particles emitted per second (particles/second)

**profile** Spacetime function that multiplies the total emission current per second.

**velocity coordinate system:** Either global or surface. A global coordinate system will specify the emission velocities according to global axis. A surface coordinate system will set the emission directions according to the normal of the emission object. So in a surface coordinate system a lower simulation bounds the emission velocity must be negative to emit into the simulation space, for an upper simulation boundary the particles must be positive to emit into the simulation space.

**mean velocity 0:** The average (mean) speed of particles in the x-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for the direction normal to the emitting surface.

**mean velocity 1:** The average (mean) speed of particles in the y-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for a direction perpendicular to the emitting surface.

**mean velocity 2:** The average (mean) speed of particles in the z-direction when *velocity coordinate system* is set to “global”. If set to “surface” then this will be the average velocity for a direction perpendicular to the emitting surface.

**thermal velocity 0:** A spread (standard deviation) for particle speeds in the 0 direction.

**thermal velocity 1:** A spread (standard deviation) for particle speeds in the 1 direction.

**thermal velocity 2:** A spread (standard deviation) for particle speeds in the 2 direction.

- **Fowler Nordheim Emission:**

Specify particle emission according to the Fowler-Nordheim model. Only available with variable or managed weight electron particle species that are not speed-limited.

**work function [eV]:** Work function of the material from which emission is occurring.

**A:** Coefficient A of the Fowler-Nordheim emission model.

**B:** Coefficient B of the Fowler-Nordheim emission model.

**field enhancement:** Multiplies the measured electric field by this amount.

**Cv:** Coefficient Cv of the Fowler-Nordheim emission model.

**Cy:** Coefficient Cy of the Fowler-Nordheim emission model

- **Richardson Dushman Emission:**

Specify particle emission according to the Richardson-Dushman model. Only available with variable or managed weight electron particle species that are not speed-limited.

**work function [eV]:** Work function of the material from which emission is occurring. Parameter in the Richardson-Dushman model.

**field evaluation offset:**

**temperature (K):** Temperature of the material from which emission is occurring. Parameter in the Richardson-Dushman model.

**field enhancement:** Multiplies the measured electric field by this amount.

**flux multiplier:** Multiplies the resulting output current by this amount.

- **Child Langmuir Emission:**

Specify particle emission according to the Child Langmuir model. Only available in simulations with variable or managed weight electron particle species that are not speed-limited.

**space charge limited emission:**

This will limit the current to provide a more consistent emission current, providing higher accuracy particularly in explosive emission cases, such as a pulsed power magnetron. For non pulsed-power simulations it is not necessary.

**mean velocity 0:** The average (mean) speed of particles in the 0 direction.

**mean velocity 1:** The average (mean) speed of particles in the 1 direction.

**mean velocity 2:** The average (mean) speed of particles in the 2 direction.

**thermal velocity 0:** A spread (standard deviation) for particle speeds in the 0 direction.

**thermal velocity 1:** A spread (standard deviation) for particle speeds in the 1 direction.

**thermal velocity 2:** A spread (standard deviation) for particle speeds in the 2 direction.

**emission offset** The distance away from the object that emitted particles are placed, as a fraction of a cell length in the signed normal direction from the emitter slab. This is useful when absorbing and emitting the same particle species from a given location.

**physical offset** The distance away from the object that emitted particles are placed, in meters, in the signed normal direction from the emitter slab. This is used when one wants to place the emission of particles farther away from the emission face than a fraction of a cell. Physical offset and emission offset add to give the total offset. For instance, if the emission surface normal is in the X-direction, then the total offset is calculated as  $\delta_t = \delta_e \Delta_X + \delta_p$ , where  $\delta_e$  is the emission offset,  $\Delta_X$  is the cell size in the X-direction,  $\delta_p$  is the physical offset.

NOTE: To allow the user to emit from non-Maxwellian probability distribution functions, one can import a Python space time function (see the “SpaceTimeFunctions” section in the Reference manual). The easiest way to do this is to right-click with your mouse on the component of the *mean velocity* you wish to emit (for example *mean velocity 0*) then click on *Assign SpaceTimeFunction* and finally choose the python *SpaceTimeFunction* you have already defined. It is best not to mix this method with the default method of choosing the velocity from a Maxwellian. Therefore, when you write your python *SpaceTimeFunction*, include all drift- and thermal-velocity terms in the Python function. Then leave the *thermal velocity 0* option set to 0.0. Also, all three components are independent. So if your python function depends only on component 0, then you could treat the other two components as Maxwellian and fill in the *mean velocity* and *thermal velocity* options using the methods discussed above for components 1 and 2.

## Secondary Electron Emitter

Secondary electrons can be emitted from a electron species, charged particle species, or neutral particle species. They can then be emitted into a separate electron species, or into the same electron species.

To specify a secondary emitter an emitter boundary type must be chosen, as well as an emission specification algorithm.

The available types of emitter boundaries are.

- **boundary emitter** This is any *Boundary Absorb and Save* boundary condition, no other specification is necessary.
- **cut cell emitter** This is any *Cut-Cell Absorb and Save* boundary condition, no other specification is necessary.
- **interior emitter** This is any *Interior Absorb and Save* Boundary Condition. It does require specification of a few more parameters.

**emission axis**

The axial direction particles will be emitted in.

**emission direction**

Either positive or negative, particles will be emitted in this direction along the emission axis.

#### **emission coordinate**

The precise coordinate at which the emission will occur. Effectively this is an offset from the position of the particle boundary condition.

4 types of emission specification are available in VSim.

- **material properties** This will emit particles according to the Furman and Pivi algorithm for either **copper** or **stainless steel**. If using variable, or managed weight particles it is also necessary to specify a minimum weight for the emitted particle.
- **constant probability** This will emit particles with a predefined probability, between 0.0 and 1.0.
- **simple**  
A secondary emitter that at most emits a single secondary electron macroparticle. See the “simpleSec” section in the Reference manual for more information.

#### **emission probability**

The probability of a particle being emitted. This is slightly different from the constant probability secondary electron emitter as this will emit only a single electron, with a prescribed probability independent of all other factors.

#### **emitted energy**

The energy of the emitted electrons if constant weight particles are used.

- **Furman and Pivi** This will emit electrons according to the Furman and Pivi algorithm. See table 1 of [FP02] For an example of how these are computed. Note this is only available for use with cut cull emitters.

#### **E0**

Total energy of the secondary electron yield, must be multiplied by ELEMCHARGE.

#### **D\_MAX**

Max Delta of the secondary electron yield.

#### **minimum weight**

Only specified if using a variable or managed weight particle species, the emitted secondary electron must be above this weight to be emitted.

## **Sputter Emitter**

Neutral particle types may sputter from other particle species.

#### **description**

A space to provide a descriptive comment of the emitter.

#### **emitter type**

The type of emitter to use, either a sputter emitter or interior sputter emitter.

- **sputter emitter**

This type of emitter should be used when emitting off a simulation particle boundary condition.

#### **particle boundary condition**

The particle boundary condition to sputter particles from. It must be of the type *Boundary Absorb and Save*. It can be a particle boundary condition from a different particle species, which will cause incident particles of that species to sputter as the neutral particle species.

#### **material properties**

This will determine the type of atom that is sputtered. Should be the same as the species to which this emitter is applied.

**sputtered velocity sigma**

The standard deviation of the velocity distribution of emitted particles.

- **interior sputter emitter**

This emitter is for use when emitting off an interior particle boundary condition.

**particle boundary condition**

The particle boundary condition to sputter particles from. It must be of the type *Interior Absorb and Save* or *Cut Cell Absorb and Save*. It can be a particle boundary condition from a different particle species, which will cause incident particles of that species to sputter as the neutral particle species.

**material properties**

This will determine the type of atom that is sputtered. Should be the same as the species to which this emitter is applied.

**sputtered velocity sigma**

The standard deviation of the velocity distribution of emitted particles.

**emission axis**

The axis along which particles are to be emitted, 0 for the 0th dimension, 1 for the 1st dimension or 2 for the 2nd dimension.

**emission direction**

Set to either positive or negative, this is the direction in which particles are emitted.

**emission coordinate**

The coordinate at which particles are actually emitted. Effectively this is an offset from the position of the particle boundary condition.

## Properties of Particle Boundary Conditions

Particle boundary conditions may be set to any particle type. If a particle boundary condition is not set on a simulation boundary, it is automatically set to absorbing.

**Absorbing:**

This type of particle boundary condition will absorb incident particles. It will not save any data from them, so they may not be used in histories or secondary emitters.

**volume**

- lower x slab
- lower y slab
- lower z slab
- upper x slab
- upper y slab
- upper z slab
- cartesian 3d slab
- cylindrical 2d slab
- index 3d slab

**Boundary Absorb and Save:**

This type of boundary condition will absorb incident particles and save them for other uses, such as histories or secondary emission.

**volume:** The simulation boundary over which to apply this condition.

**Boundary Accumulate:**

Incident particles will be accumulated on this boundary and not allowed to move. This allows for the charging of a simulation boundary. The accumulated particles will be stored in a new particle species, appended with the prefix heavy.

**volume:** The simulation boundary to accumulate particles on.

**Boundary Diffuse Reflector:**

This type of particle boundary condition will reflect particles back into the simulation space, with an user supplied velocity. It may only be applied on simulation boundaries.

- **average velocity 0**
- **average velocity 1**
- **average velocity 2**
- **thermal velocity 0**
- **thermal velocity 1**
- **thermal velocity 2**
- **volume**
  - lower x slab**
  - lower y slab**
  - lower z slab**
  - upper x slab**
  - upper y slab**
  - upper z slab**
  - lower r slab**
  - upper r slab**

**Cut-Cell Absorb and Save:**

This type of boundary condition will absorb incident particles and save them for other uses, such as histories or secondary emission. It may be applied to geometries. In VSim9.0, only one Cut-Cell Absorb and Save boundary condition may be used per particle species. If this boundary condition is used Interior Absorb and Save boundary conditions may not be used.

**volume**

- **object name**  
The name of the geometry over which to apply this condition.

**use relative velocity**

If set to False, then this saves an Absorbed Particle Log velocity history in the simulation coordinate system.  
If set to True, then the velocity history data are saved in a coordinate systems such that component 1 is normal to the geometry surface, and components 2 and 3 are each tangent to the component surface

**Cut-Cell Accumulate:**

Incident particles will be accumulated on this boundary and not allowed to move. This allows for the charging of a shape The accumulated particles will be stored in a new particle species, appended with the prefix heavy.

**volume:** The shape to accumulate particles on.

**use relative velocity**

If set to False, then this saves an Absorbed Particle Log velocity history in the simulation coordinate system.  
If set to True, then the velocity history data are saved in a coordinate systems such that component 1 is normal to the geometry surface, and components 2 and 3 are each tangent to the component surface

**Interior Absorb and Save:**

This type of boundary condition will absorb incident particles and save them for use with histories or secondary emission. It may be set to an internal volume of the simulation space.

**volume:** Depending on simulation properties this will be either a:

- **cylindrical 2D slab**
- **cartesian 3D slab**

**Interior Accumulate:**

Incident particles will be accumulated on this boundary and not allowed to move. This allows for the charging of a user specified interior plane. The accumulated particles will be stored in a new particle species, appended with the prefix heavy.

**volume**

**xMin**

The minimum x position of the boundary condition.

**xMax**

The maximum x position of the boundary condition.

**yMin**

The minimum y position of the boundary condition.

**yMax**

The maximum y position of the boundary condition.

**zMin**

The minimum z position of the boundary condition.

**zMax**

The maximum z position of the boundary condition.

**surface**

The surface of the described volume to accumulate particles on.

**Interior Diffuse Reflector:**

This type of particle boundary condition will reflect particles back into the simulation space, with an user supplied velocity. It may only be applied on internal planes of the simulation space

**average velocity 0**

**average velocity 1**

**average velocity 2**

**thermal velocity 0**

**thermal velocity 1**

**thermal velocity 2**

**orthogonal direction**

Incoming direction of particles to apply the boundary condition on. Set as negative if particles are coming from the negative direction and positive if particles are coming from the positive direction

**negative x**

**x coordinate**

x coordinate of the boundary condition.

**lower y coordinate**

lower y coordinate of the boundary condition.

**lower z coordinate**

lower z coordinate of the boundary condition.

**upper y coordinate**

upper y coordinate of the boundary condition.

**upper z coordinate**

upper z coordinate of the boundary condition.

**negative y**

**y coordinate**

y coordinate of the boundary condition.

**lower x coordinate**

lower x coordinate of the boundary condition.

**lower z coordinate**

lower z coordinate of the boundary condition.

**upper x coordinate**

upper x coordinate of the boundary condition.

**upper z coordinate**

upper z coordinate of the boundary condition.

**negative z**

**z coordinate**

z coordinate of the boundary condition.

**lower x coordinate**

lower x coordinate of the boundary condition.

**lower y coordinate**

lower y coordinate of the boundary condition.

**upper x coordinate**

upper x coordinate of the boundary condition.

**upper y coordinate**

upper y coordinate of the boundary condition.

**positive x**

**x coordinate**

x coordinate of the boundary condition.

**lower y coordinate**

lower y coordinate of the boundary condition.

**lower z coordinate**

lower z coordinate of the boundary condition.

**upper y coordinate**

upper y coordinate of the boundary condition.

**upper z coordinate**

upper z coordinate of the boundary condition.



**positive y****y coordinate**

y coordinate of the boundary condition.

**lower x coordinate**

lower x coordinate of the boundary condition.

**lower z coordinate**

lower z coordinate of the boundary condition.

**upper x coordinate**

upper x coordinate of the boundary condition.

**upper z coordinate**

upper z coordinate of the boundary condition.

**positive z****z coordinate**

z coordinate of the boundary condition.

**lower x coordinate**

lower x coordinate of the boundary condition.

**lower y coordinate**

lower y coordinate of the boundary condition.

**upper x coordinate**

upper x coordinate of the boundary condition.

**upper y coordinate**

upper y coordinate of the boundary condition.

**Interior Partial Transmitter:**

An interior partial transmitter will transmit a user specified fraction of the incident particles and either absorb or reflect the remainder. If reflected the user can specify the velocity with which they are reflected. This is a *one way* particle boundary condition, as set by the orthogonal direction.

**orthogonal direction:** Incoming direction of particles to apply the boundary condition on.

**negative x****x coordinate**

x coordinate of the boundary condition.

**lower y coordinate**

lower y coordinate of the boundary condition.

**lower z coordinate**

lower z coordinate of the boundary condition.

**upper y coordinate**

upper y coordinate of the boundary condition.

**upper z coordinate**

upper z coordinate of the boundary condition.

**negative y****y coordinate**

y coordinate of the boundary condition.

**lower x coordinate**

lower x coordinate of the boundary condition.

**lower z coordinate**

lower z coordinate of the boundary condition.

**upper x coordinate**

upper x coordinate of the boundary condition.

**upper z coordinate**

upper z coordinate of the boundary condition.

**negative z**

**z coordinate**

z coordinate of the boundary condition.

**lower x coordinate**

lower x coordinate of the boundary condition.

**lower y coordinate**

lower y coordinate of the boundary condition.

**upper x coordinate**

upper x coordinate of the boundary condition.

**upper y coordinate**

upper y coordinate of the boundary condition.

**positive x**

**x coordinate**

x coordinate of the boundary condition.

**lower y coordinate**

lower y coordinate of the boundary condition.

**lower z coordinate**

lower z coordinate of the boundary condition.

**upper y coordinate**

upper y coordinate of the boundary condition.

**upper z coordinate**

upper z coordinate of the boundary condition.

**positive y**

**y coordinate**

y coordinate of the boundary condition.

**lower x coordinate**

lower x coordinate of the boundary condition.

**lower z coordinate**

lower z coordinate of the boundary condition.

**upper x coordinate**

upper x coordinate of the boundary condition.

**upper z coordinate**

upper z coordinate of the boundary condition.

**positive z**

**z coordinate**

z coordinate of the boundary condition.

**lower x coordinate**

lower x coordinate of the boundary condition.

**lower y coordinate**

lower y coordinate of the boundary condition.

**upper x coordinate**

upper x coordinate of the boundary condition.

**upper y coordinate**

upper y coordinate of the boundary condition.

**non transmitted particles****particles absorbed****particles reflected**

Note that the velocity corresponding to the orthogonal direction should always be specified as a positive number. If it is not this will be automatically corrected. **average velocity 0**

**average velocity 1****average velocity 2****thermal velocity 0****thermal velocity 1****thermal velocity 2****Specular Reflecting:**

This type of particle boundary condition will reflect particles back into the simulation space. It may only be applied on simulation boundaries.

**volume**

- **lower x slab**
- **lower y slab**
- **lower z slab**
- **upper x slab**
- **upper y slab**
- **upper z slab**
- **lower r slab**
- **upper r slab**

**Particle Loader**

All kinetic particles have access to the same type of particle loader.

**load density**

Whether to define the density of loaded particles by macroparticles or physical particles.

**relative density**

Use a function defining the density of loaded particles in macroparticles. With constant weight particles this is a value between 0 and 1 that will be multiplied by the macroparticles per cell of the particles definition.

**physical density**

Use a function defining the density of loaded particles in physical particles. With variable weight particles this

value is divided by the *nominal density* of the particles definition and is used to modify the weight of each loaded macroparticle.

#### **particle load placement**

Whether to use a bit-reversed or grid-defined placement of macroparticles in each cell.

- **grid**  
Place particles equidistant on the grid lines of the simulation. Useful for starting simulations “cold”.
- **macro particles per direction**  
A vector describing the number of macroparticles to load per cell on each axis.
- **bit-reversed**  
Places particles according to a bit-reversed algorithm. Number of macroparticles loaded per cell is based on the value given in the *weight setting* of the particles definition.

#### **load duration**

Whether to load the particles only at the beginning of the simulation or repeatedly.

- **initialize only**  
Particles will only be loaded at the beginning of the simulation.
- **repeat loading**  
Particles will be loaded according to the parameters below.
  - **start time**  
The time at which to start loading particles.
  - **stop time**  
The time at which to stop loading particles.
  - **load after initialization**  
Loading will repeat in all cells during the loading period.
  - **load upon shift**  
Load particles into cells brought into the simulation by a moving window.

#### **velocity distribution**

The velocity components of particles as they are loaded.

- u0**  
The velocity in the 0th dimension.
- u1**  
The velocity in the 1st dimension.
- u2**  
The velocity in the 2nd dimension.

#### **volume**

The volume in which to load particles:

- **cartesian 3d slab**
- **cylindrical 2d slab**

## Particle Loader from File Properties

All kinetic particles have access to the particle loader from file. This loader will use a user supplied .dat file consisting of particle positions and velocities. This file should have at least six columns for 3D constant weight particles, separated by spaces (no commas). For constant weight particles, the six columns specify  $x, y, z, P_x, P_y, P_z$ , where  $P_i = \gamma v_i$ . For variable weight particles, a 7th column needs to be included to specify the weight of the particle. In 2D, the user should only supply  $x, y, P_x, P_y, P_z$  for constant weight particles and a 6th column for the weight if particles are assigned a variable weight. Likewise, in 1D, the user should only supply  $x, P_x, P_y, P_z$  and an optional 5th column for variable weight particles. If the user supplies the weight (the last column) for constant weight particles, Vorpak will read in the data but the output will not be correct. Therefore, if the particles are constant weight, then be sure to only supply 6 columns in 3D, 5 columns in 2D, and 4 columns in 1D.

This file may then have a constant shift or custom density function applied to it, and can be reloaded multiple times during the simulation run.

### file

Filename to load from. Must be located in the simulation directory.

### particle shift in direction 0

A shift to apply to the 0th component of all loaded particles.

### particle shift in direction 1

A shift to apply to the 1st component of all loaded particles.

### particle shift in direction 2

A shift to apply to the 2nd component of all loaded particles.

### density function

A space time function which can modify the density of loaded particles.

### load period [timesteps]

If zero, particles only loaded at initialization, otherwise will load according to a specified period.

## 8.17.2 Fluids

### Fluids

Neutral Fluids can be added to the simulation. To add a fluid, right click the Fluids button and select *Add Fluid*

A fluid is a volume distribution where you can provide the min and max values in each direction. A fluid will show up on the *3D View*. You can hide a fluid by unchecking the box next to it. Hiding a fluid will not remove it from the simulation, just hide it in the 3D view. See [Fig. 8.26](#).

### Fluids Properties

**kind** (not editable): Neutral Fluid

**fluid temperature (K)**: The temperature of the neutral fluid in Kelvin.

**molecule**: Molecule of the neutral particle. A custom molecule is available for those not pre-defined.

- **H**: Hydrogen (atomic)
- **H2**: Hydrogen (molecular)
- **He**: Helium
- **Ar**: Argon

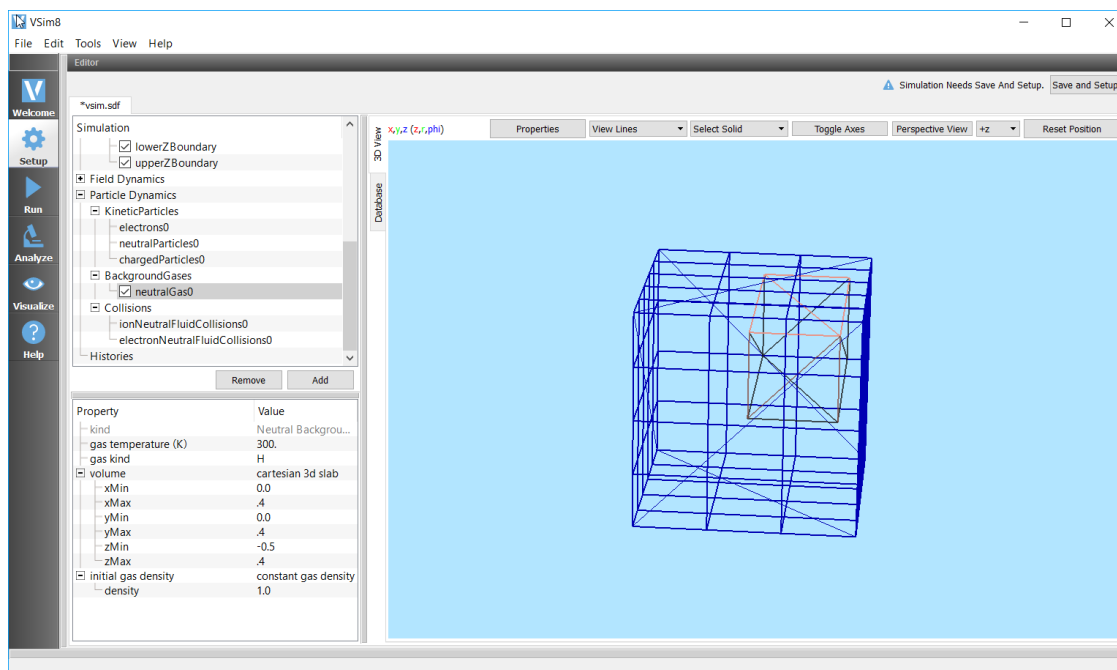


Fig. 8.26: A volume of fluid is shown in the smaller/interior box. The outer box is the grid.

- **Xe**: Xenon
- **Rn**: Radon
- **Kr**: Krypton
- **O**: Oxygen (atomic)
- **O2**: Oxygen (molecular)
- **Ne**: Neon
- **N**: Nitrogen (atomic)
- **N2**: Nitrogen (molecular)
- **custom molecule**: Set the mass (in amu) and ionization energy (in eV) of a custom gas.
  - **mass [amu]**: The mass of a single real particle in atomic mass units.
  - **ionization energy [eV]**: The ionization energy of the molecule in electron volts.

**volume**: This volume value will change depending on your dimensionality and coordinate system.

**xMin**: The minimum x position of the background fluid.

**xMax**: The maximum x position of the background fluid.

**yMin**: The minimum y position of the background fluid.

**yMax**: The maximum y position of the background fluid.

**zMin**: The minimum z position of the background fluid.

**zMax**: The maximum z position of the background fluid.

**fluid density**: Whether to use a constant or variable fluid density. A constant density will not change due to reactions with particles, while a variable will.

- **constant fluid density:** A constant fluid density is used to keep the fluid density constant after reactions with particles.
- **density:** The value of the density of the neutral fluid per cubic meter.
- **variable fluid density:** A variable fluid density is used to allow the fluid density to vary due to reactions with particles.
- **density:** The value of the density of the neutral fluid per cubic meter.

## 8.18 Collisions

There are three frameworks for setting up particle collisions available in VSim. The newest, most flexible, and fastest is the *Reactions* framework. The *Reactions* framework supplants the *Monte Carlo Interactions* framework. The *Impact Collider* (called “ReducedCollisions” in the Visual Setup) framework is the oldest framework, and is limited to interactions between kinetic particles with a neutral fluid, but runs very quickly.

In the Visual Setup, only one framework can be used at a time.





**PREPARE TAB**



## RUN TAB (EXECUTING THE COMPUTATIONAL ENGINE VORPAL)

### 10.1 Running Vorpall within GSimComposer

#### 10.1.1 Run Window

##### Select the Run Icon

Once your validation is successful, in the upper right corner you will see a checkmark by the *Save and Setup* button. You can now select the **Run** icon from the icon panel on the far left of the GSimComposer window.

Click on the **Run** icon as shown in *Run Icon in Icon Panel*.

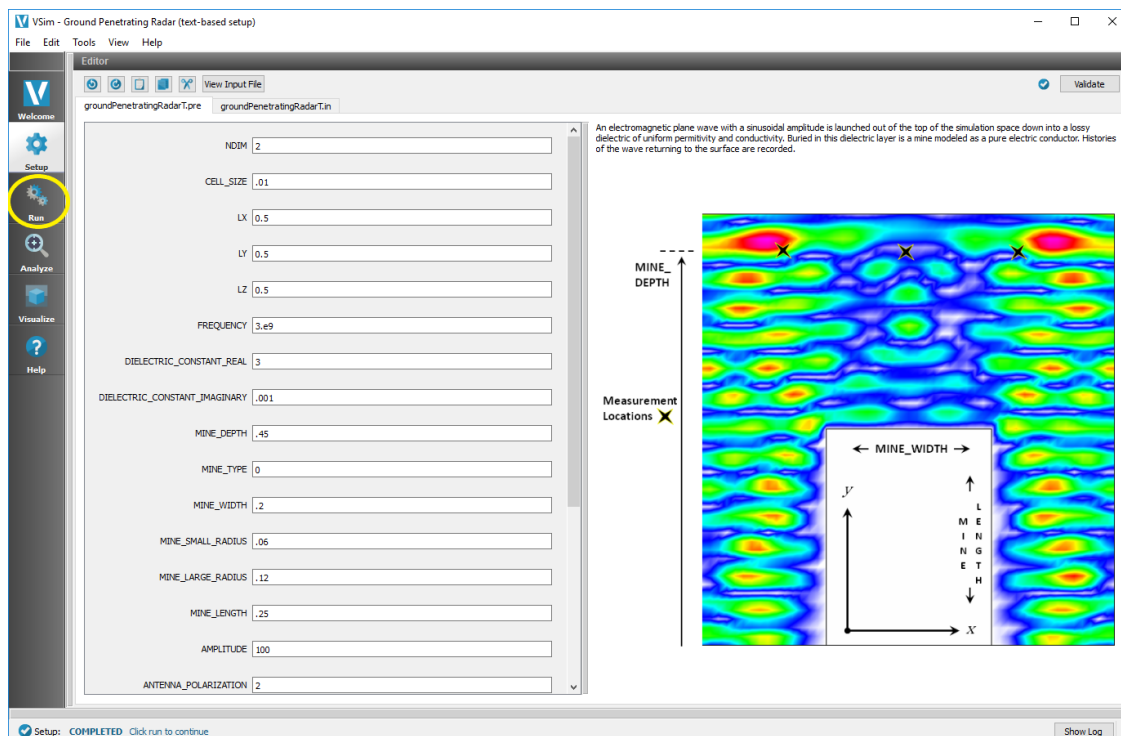


Fig. 10.1: Run Icon in Icon Panel

## The Run Window

The GSimComposer Run window contains two panes. As displayed in *The Run Window*, the *Runtime Options* pane is on the left and the *Logs and Output Files* pane is on the right, which contains an *Engine Log* and a *File Browser* tab.

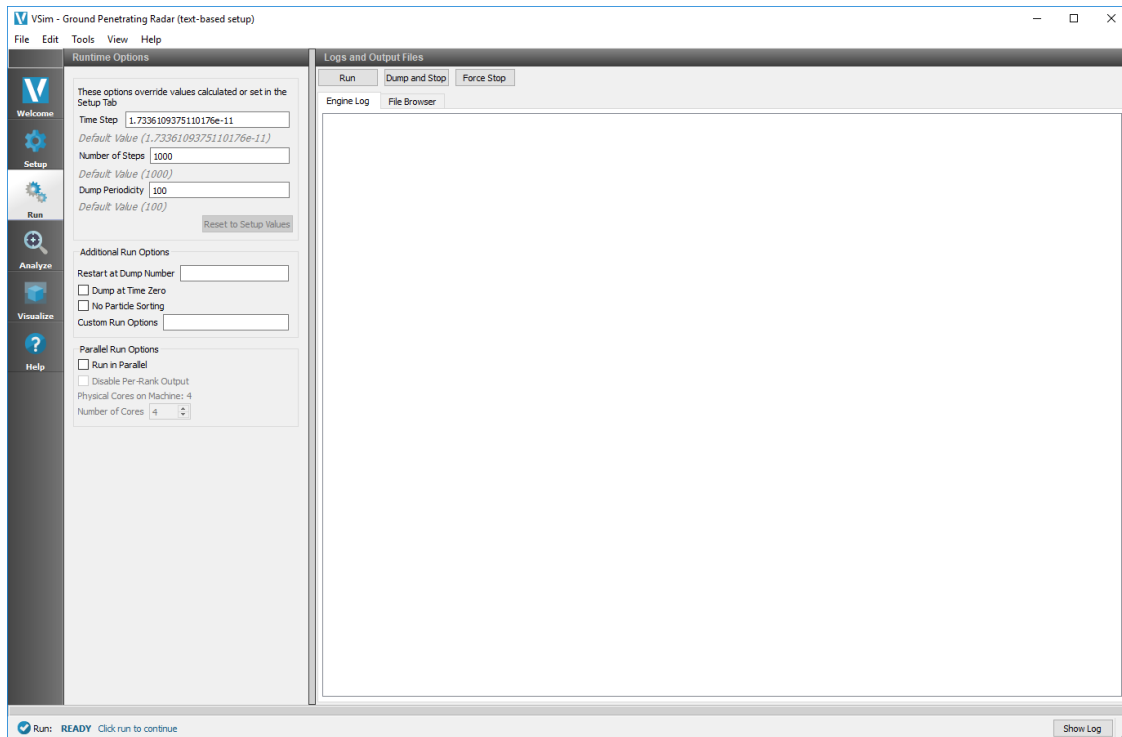


Fig. 10.2: The Run Window

## Runtime Options

GSimComposer enables you to specify runtime options, and in some cases, override the settings in your simulation input file. The *Runtime Options* pane contains fields and options that give you the flexibility of command line control with the convenience of a graphical interface much like the key input parameters of the setup window.

### Runtime Options: Parameters

#### Time Step (s):

Step size to use in the simulation. When choosing the step size for your simulation, you must consider stability requirements. For example, for electromagnetic simulations, you should specify a step size that satisfies the Courant condition [CFL28]. You can override the **dt** variable using the **-dt** command line option. You must define *dt* in an input file.

#### Number of Time Steps:

Number of steps to take. (In the case of a restart, **nsteps** is the number of additional steps.) This can be overridden with the **-n** command line parameter. You must define *nsteps* in an input file or on the command line.

#### Dump Periodicity (time steps):

How often to dump the data; indicates data is to be dumped whenever the time step has increased by this amount. The command line parameter **-d** overrides this variable. Must have this defined in an input file or on the command line.

**Restart at Dump Number:**

It is possible in this menu to restart a previous simulation with the *Restart at Dump Number* field. VSim will load all the data associated with the dump number input in the field, and continue running the simulation from that time.

**Dump at Time Zero:**

Dump data at start of simulation, before any time has passed. This option is useful for debugging purposes. It lets you see whether Vorpall used the data you wanted it to use at the start of the simulation.

**No Particle Sorting:**

This passes the `-ns` flag to Vorpall, which tells Vorpall to not sort the particles. Sorting the particles (does not work with cell species) can affect the performance of your simulation.

**Custom Run Options:**

You can pass command line arguments to the Vorpall engine here. The list of options can be found in the Reference Manual under Vorpall Command Line Options.

If you make changes to the *Runtime Options*, you can restore the options to their original settings by clicking on the *Reset to Setup Values* button located below the *Dump Periodicity* in the left pane.

## Runtime Options: Run Mode

There are 3 modes for running: Serial, Parallel, and Scheduler. There are different options for the different modes described below.

**GPU Options (Serial/Parallel/Scheduler):**

If you have a GPU license then you may run your simulation on one or more GPU cards based on the number of SMs of the cards. For serial runs you are only allowed to run on one GPU card. For parallel runs the number of processes should match the number of GPU cards in the run. For scheduler runs it left to user to pick the number of GPUs and processes per task to match what is on the system.

**Number of Processes (Parallel):**

Select the number of processes (or ranks) to run in parallel by entering an integer here. For convenience, the number of *Physical Cores on Machine* and *Licensed Cores* are shown. You cannot run on more cores than you are licensed for. The contents of the *Engine Log* are dumped into separate comms files when you run in parallel and the log shown is for the 0th MPI Rank. There are places to enter an MPI host file and other options if needed, but these are generally not needed. Additional information on running in parallel can be found below in [Running in Parallel from Composer](#).

**Scheduler Options:**

There are options in the Scheduler mode that are specific to the particular scheduler software chosen in the *Scheduler:* option. If there are no available choices in the *Scheduler:* option, then Composer has not been able to detect an available scheduler. The supported schedulers are: Slurm, LSF, OpenPBS, & SGE.

## Run the Simulation

For our example, we'll run this simulation using only the default existing settings from the input file.

You do not need to select any file in particular in the *File Browser* tab before clicking on the *Run* button. However, if the *File Browser* tab display area is too narrow for you to see the full file names in the filename list and you would like to see the file name extensions of the files in the file browser, you can adjust the width of the filename field by using your mouse to drag the column border.

Click on the *Run* button at the top of the *Logs and Output Files* pane as shown in [Run Button](#).

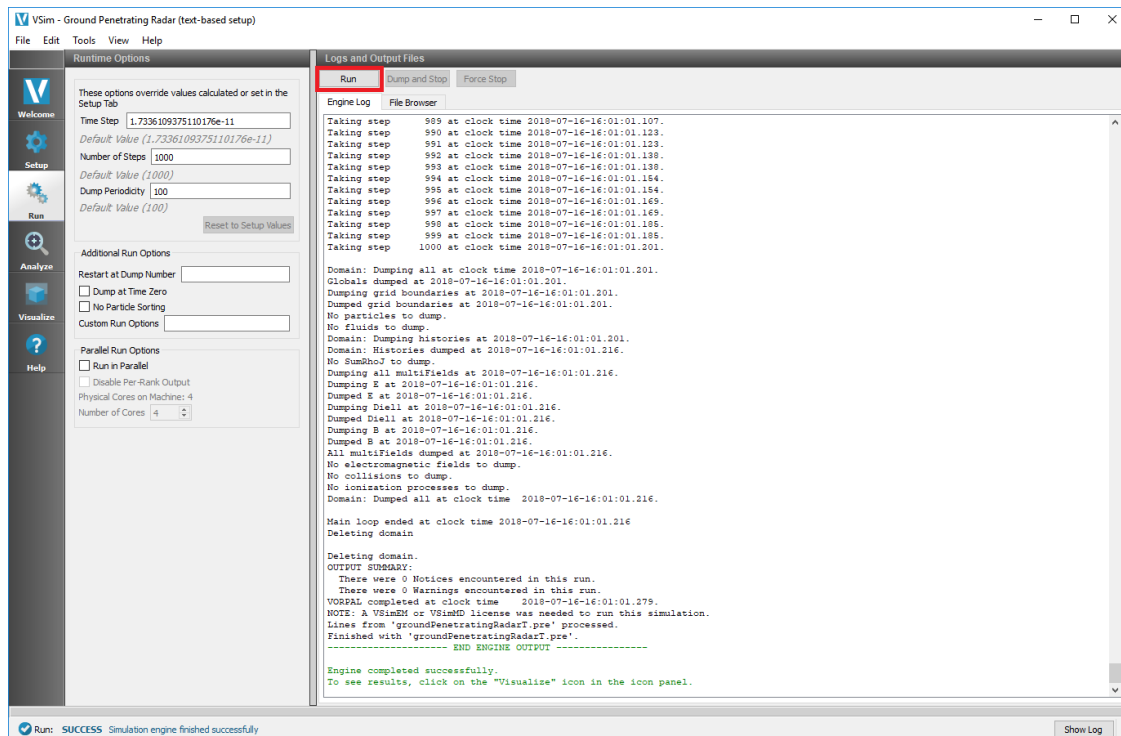


Fig. 10.3: Run Button

## Stopping the Simulation

GSimComposer features the ability to *Dump and Stop* and *Force Stop* a simulation. The buttons for these actions are located next to the *Run* button. The two actions have slightly different uses. The *Dump and Stop* button is to halt a simulation that is running normally to free up the processors used for another task, or so that one may vary the parameters and restart. When a simulation is stopped it will dump all field and history data, so that it may be restarted from the same point later. The output of a successfully stopped simulation is shown in [Stopped Simulation](#).

The *Force Stop* is to be used if you realize that an error was made in the input file after clicking *Run* and needs to be corrected. If *Force Stop* is used the field and history data will *NOT* be written to a .h5 file before the simulation stops, but it will stop the simulation immediately rather than exiting gracefully. This option is particularly useful when you may have more mesh cells than you intended, where the simulation is trying to allocate more memory than you intended. The output of a successfully force stopped simulation is given below.

## Restarting a Simulation

With GSimComposer it is possible to restart a simulation that has previously been run. This is useful if it is desired to add more time steps to the initial simulation, or if the simulation had been stopped in the middle of the run. Underneath the *Runtime Options* pane of the run window, under *Additional Run Options*, there is a *Restart at Dump Number* field.

Simply put in the last memory dump of the simulation and then click on the *Run* button, like running a normal simulation. This process is demonstrated in [Restarting a Simulation](#).

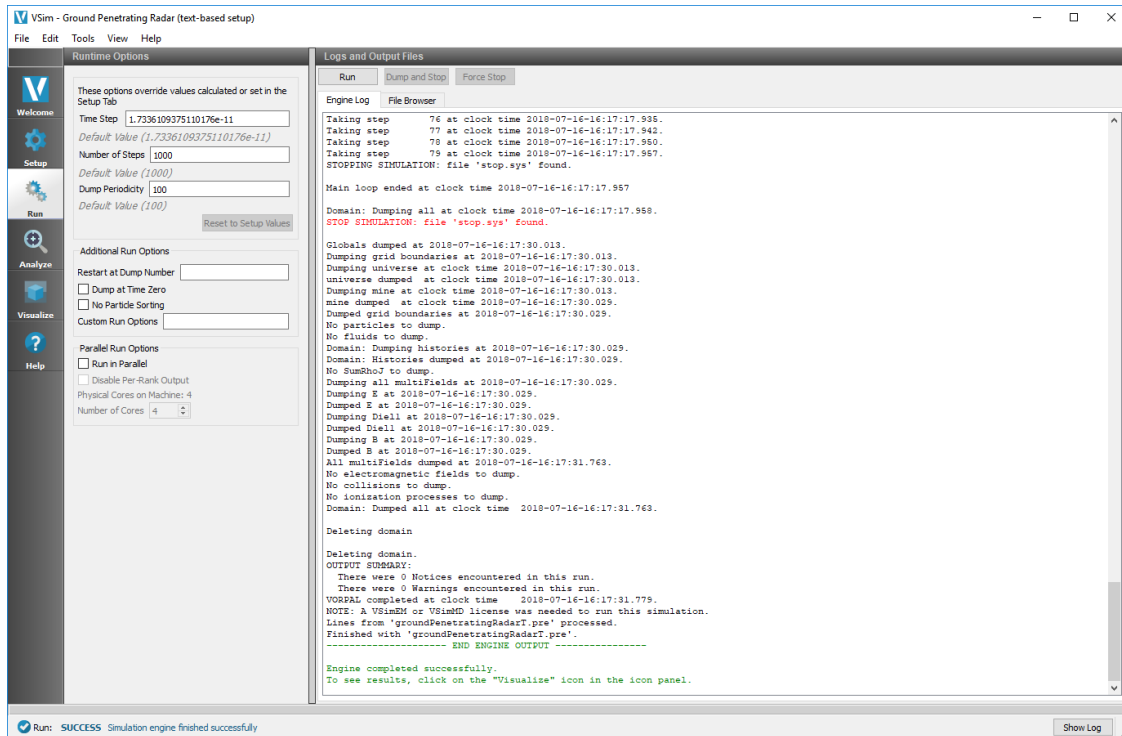


Fig. 10.4: Stopped Simulation

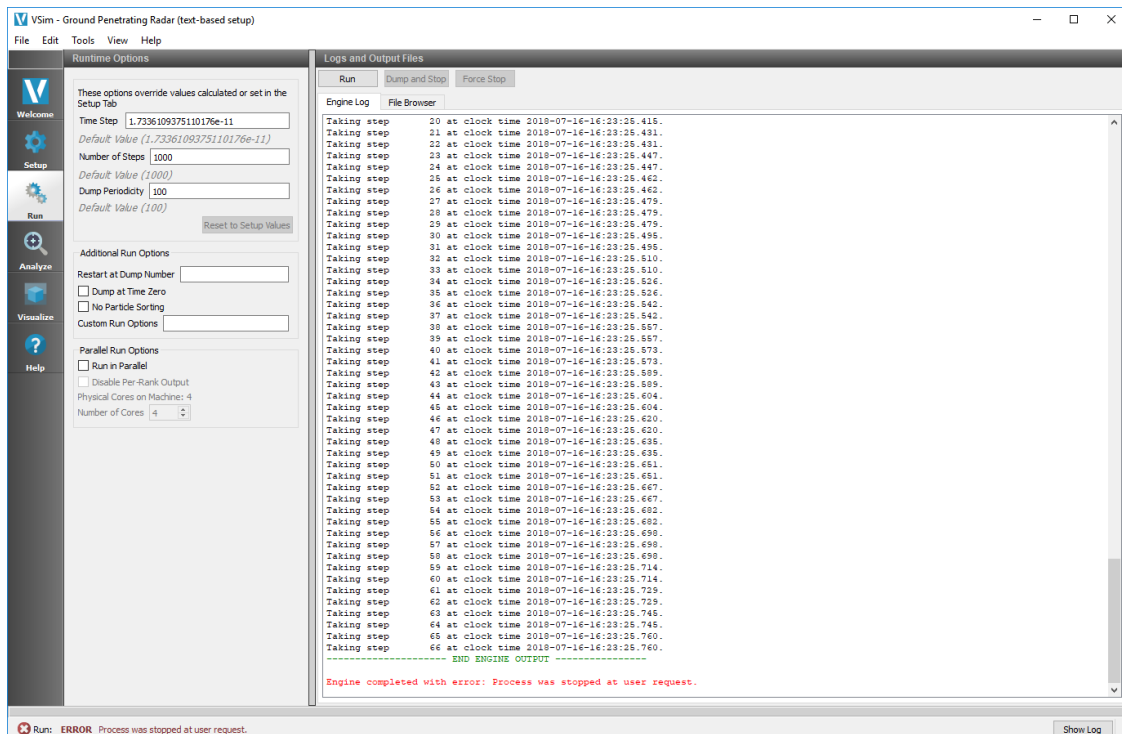


Fig. 10.5: Force Stopped Simulation

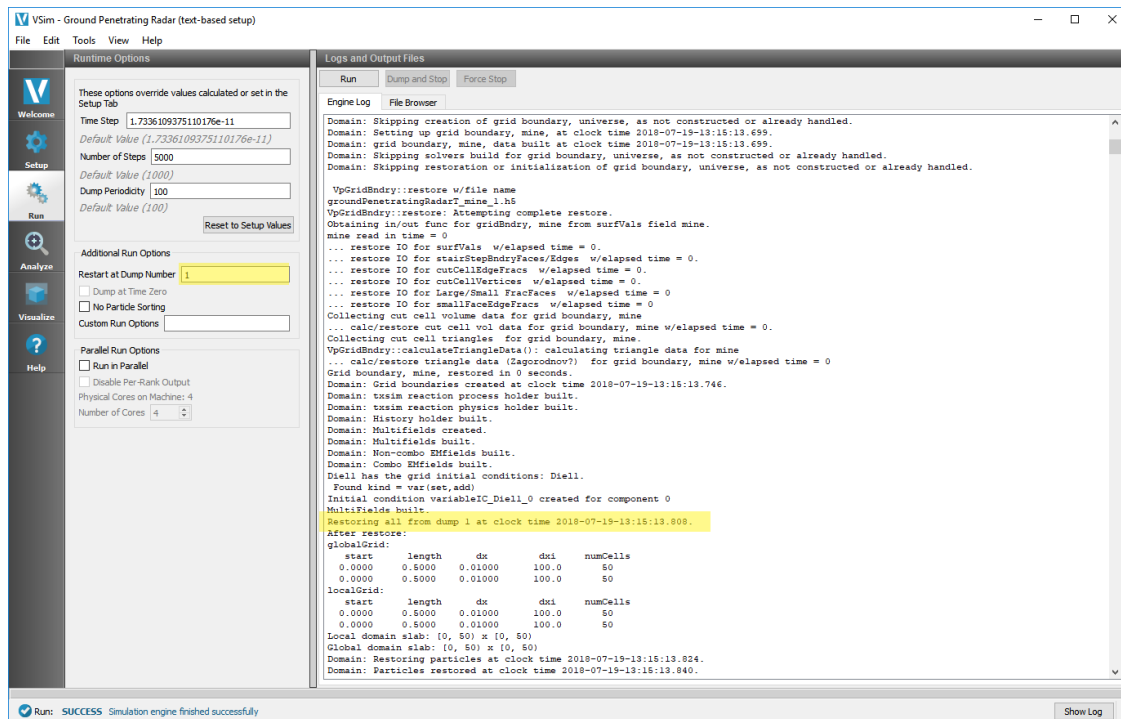


Fig. 10.6: Restarting a Simulation

## View the Engine Log

GSimComposer notifies you of the progress of its activity by reporting results along the way in the *Engine Log* tab as shown in *Engine Log*.

## File Browser Tab in the Logs and Output Files Pane

In previous steps, the *File Browser* tab was located behind the *Engine Log* tab in the *Logs and Output Files* pane. Click on the *File Browser* tab to bring it to the front as shown in *File Browser Tab in Logs and Output Files Pane*.

As with the *File Browser* in the *Setup* window, the *File Browser* in the **Run** window also has the *Smart Grouping* and *All Files* pull-down menus at the bottom of the tab.

After a simulation has been run, you will be able to see the files that were output based on your number of time steps and the dump periodicity. See *File Browser Tab in Logs and Output Files Pane*.

## Output File Naming Conventions

The first part of the output file name is the name of the input file. This is often referred to as the *base name*.

The second part of the output file name indicates the file's contents, for example:

- The name of the field or particle species, such as E, B, or electrons.
- Globals for the file containing global variables such as the global grid, needed for restarts.
- History, containing data recorded over time.
- comms and "all" text files, containing various debugging and information about the simulation.



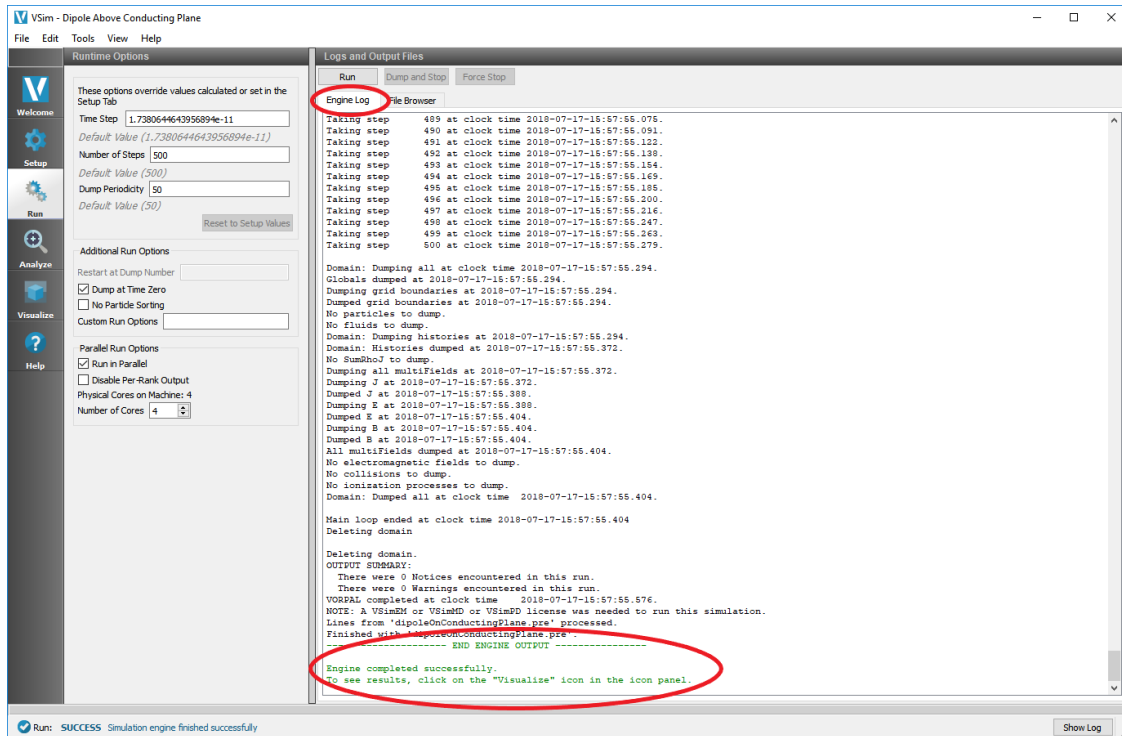


Fig. 10.7: Engine Log

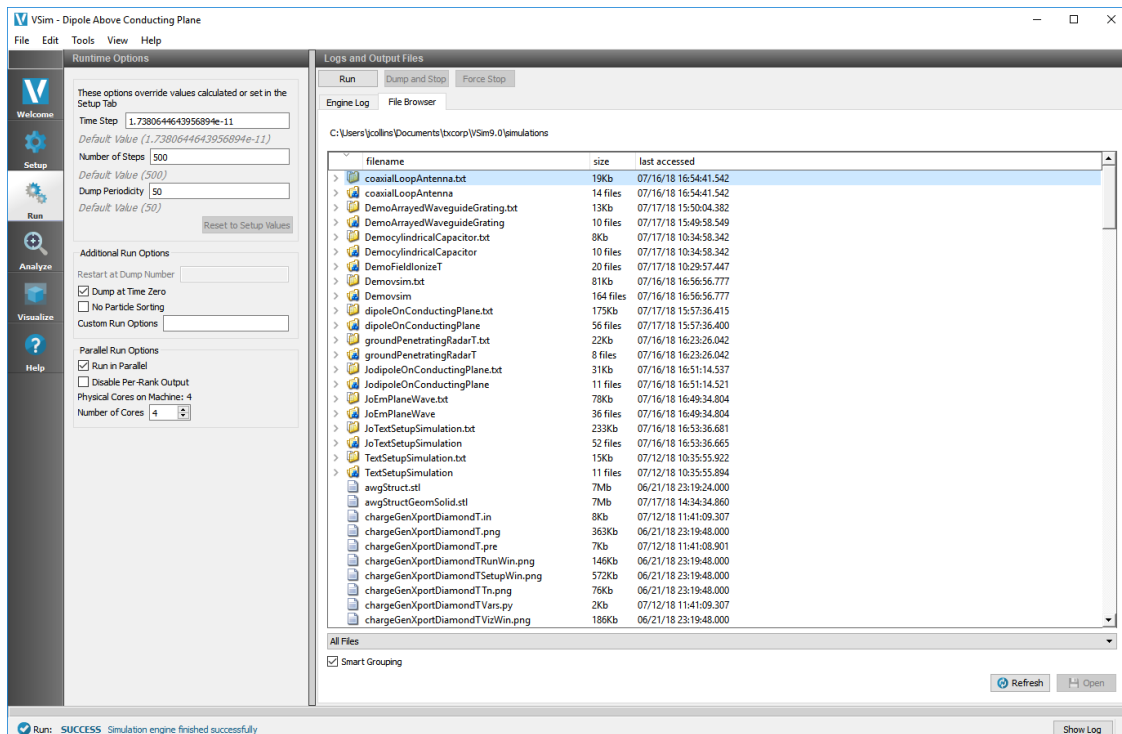


Fig. 10.8: File Browser Tab in Logs and Output Files Pane

The third part of the output file name is the dump number.

The final part of the output file name is the suffix.

### 10.1.2 Running in Parallel from GSimComposer

GSimComposer runs simulations in serial by default when you open a new simulation. If you are running on a local system with multiple cores, you can run your simulation in parallel as multiple processes.

#### Permanently Switching the Engine to Parallel Execution

One can switch the engine to parallel execution from within *Tools -> Settings*, by following the steps provided in [MPI](#). See *Switching on parallel execution in the settings tab*.

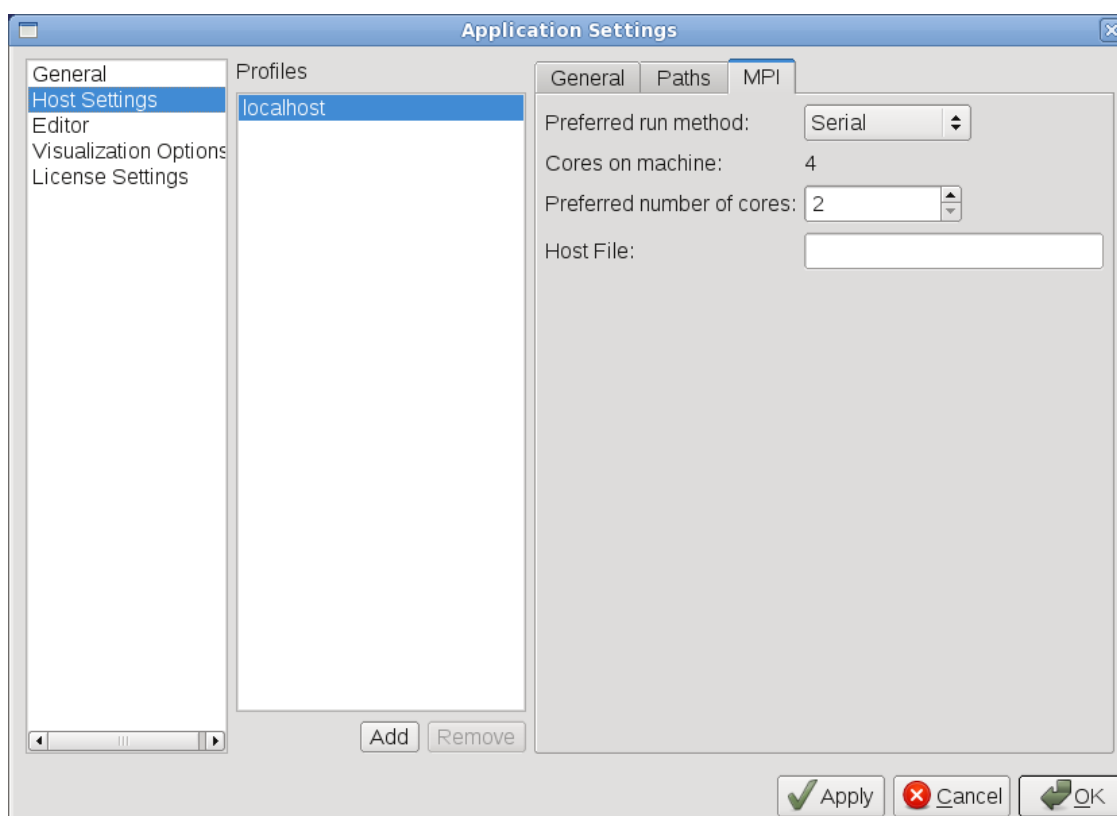


Fig. 10.9: Switching on parallel execution in the settings tab.

## Defining the Number of Processors From the Run Window

It is also possible to switch the number of processors the simulation is run on within a single simulation session in the *Run* window. In the *Runtime Options* pane, you will find the *Parallel Run Options* box. Here you can define the number of cores to run on. You cannot use more cores than you are licensed for. The options will appear with their defaulted values for that simulation, but you can override the defaults. This setting will be retained for as long as the current file is open. See [Changing number of cores on a per-simulation basis](#).

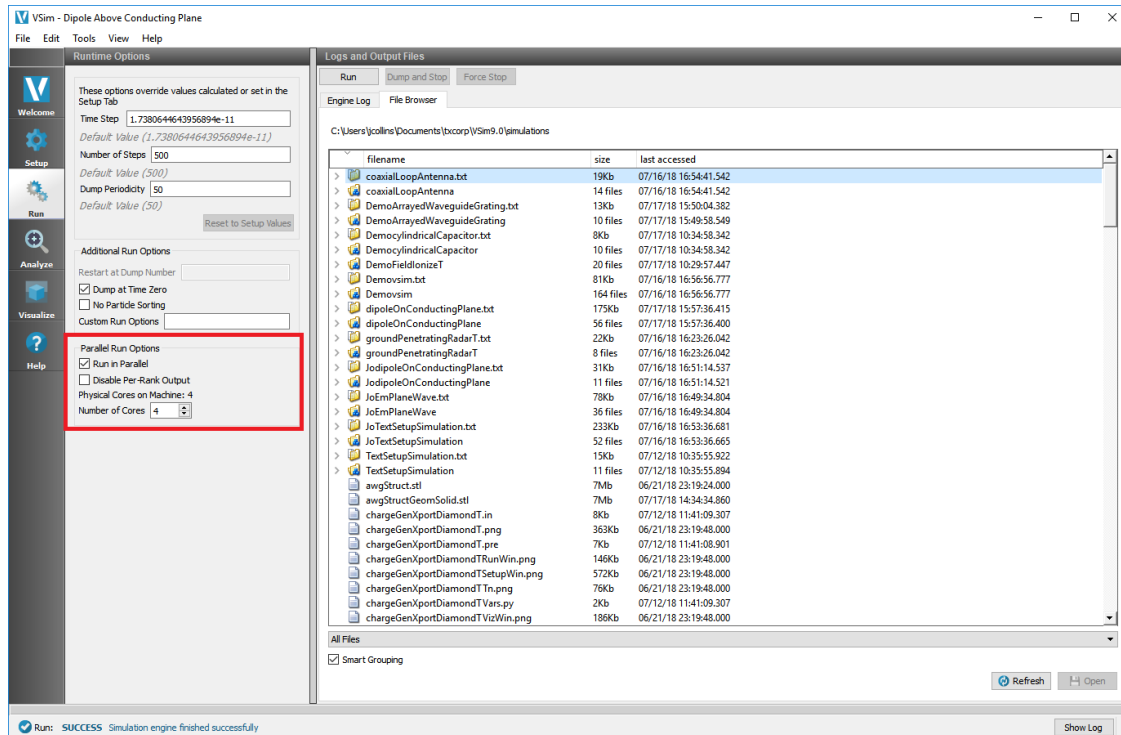


Fig. 10.10: Changing number of cores on a per-simulation basis

Now change any command line options as desired or run as usual by pressing the *Run* button.

### 10.1.3 Running via Scheduler from GSimComposer

GSimComposer runs simulations in serial by default when you open a new simulation.

## Defining the Scheduler Options From the Run Window

It is also possible to switch the number of processors the simulation is run on within a single simulation session in the *Run* window. In the *Runtime Options* pane, you will find the *Scheduler* box. Here you can define the number of cores to run on. You cannot use more cores than you are licensed for. The options will appear with their defaulted values for that simulation, but you can override the defaults. This setting will be retained for as long as the current file is open. See [Scheduler Options](#).

Now change any command line options as desired or run as usual by pressing the *Run* button.

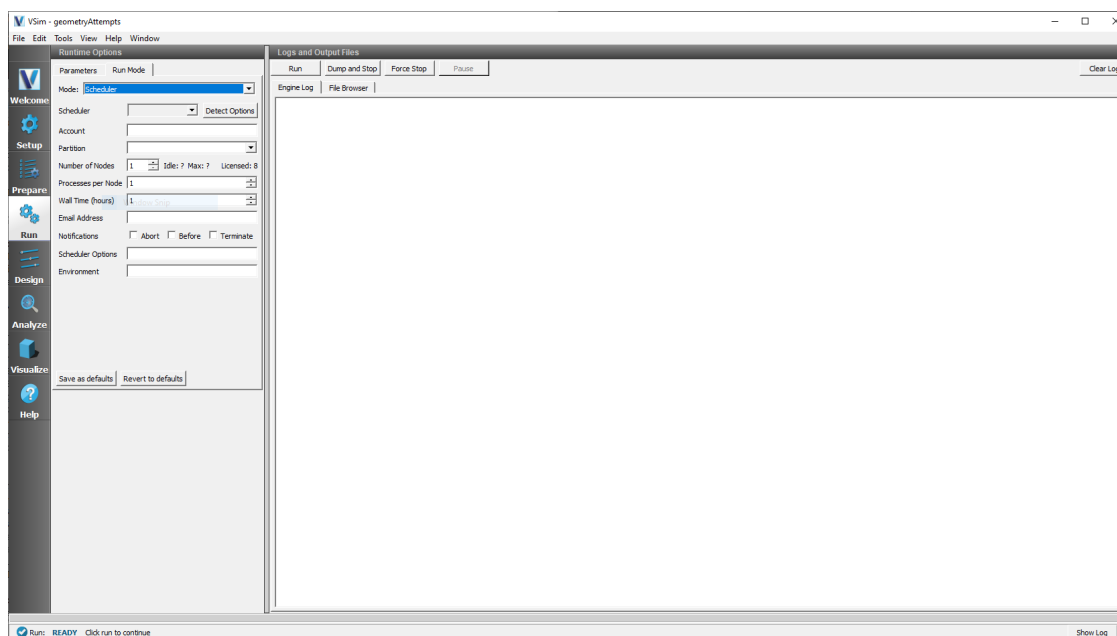


Fig. 10.11: Scheduler Options

## 10.2 Running Vorpall from the Command Line

### 10.2.1 Running Vorpall From the Command Line

The following sections describe how to run Vorpall from the command line.

#### Setting Up Vorpall Command Line Environment

Vorpall needs several environment variables set before it can be run from the command line. GSim provides scripts to setup the environment on each operating system.

The following instructions use the variable `SCRIPT_DIR` which is the directory where GSim is installed. For example, this would be something like

```
SCRIPT_DIR=C:\Program Files\Tech-X\GSim1.0 (Windows),
SCRIPT_DIR=/usr/bin/GSim1.0 (Linux),
SCRIPT_DIR=/Applications/GSim1.0/GSimComposer.app/Contents/Resources (Mac).
```

#### On Windows

Open a Command Prompt (run `cmd.exe`) and execute the following line:

```
C:\> %SCRIPT_DIR%\setupCmdEnv.bat
```

## On Linux or Mac

In a bash shell, source the GSimComposer.sh script as follows:

```
$ source $SCRIPT_DIR/GSimComposer.sh
```

This is a bash shell script, which means you must be running the bash shell to execute the above command. If you are normally a csh/tsh user, you will need to start up a bash shell to execute the above command and subsequently execute GSimComposer.

Most distribution's operating systems will allow you to add the above command to the .bashrc file in your home directory, which will prevent having to run it each time you log in. Any changes you make to your .bashrc do not take effect until the next time you log in, so after modifying your .bashrc file, you must execute the following command in your current shell, but will not need to do it in the future:

```
$ source ~/.bashrc
```

## Serial Computation

The Vorpall executable for use in serial computation is named **vorpalsr**. Except as noted, the explanations and tutorials within the “User Guide” and “Example” Manuals demonstrate Vorpall usage for serial computations. Here is an example of Vorpall command line invocation using an input file named `myfile.pre` and specifying 1000 time steps, outputting the result data (dumping) every 500 steps. By default, the output files for this example would be named using the format `myfile.out`.

```
vorpalsr -i myfile.pre -n 1000 -d 500
```

The Vorpall computation engine for serial computations also creates a single text file named `myfile_comms_0.txt` unless this has been suppressed by command line or input file options.

## Parallel Computation

The Vorpall executable for use in parallel computation is named **vorpall**. This section explains use of the Vorpall executable program for parallel computations.

Vorpall for parallel computations requires the Message Passing Interface (MPI). On Mac and Windows, you must use our bundled MPI. On Windows the parallel message passing interface (MPI) library provided with GSim is MS MPI (from Microsoft). On Mac, the parallel message passing interface (MPI) library provided with GSim is OpenMPI. On Linux, the parallel message passing interface (MPI) library provided with GSim is MPICH. If there is a reason why you must use a system **MPI**, please contact Tech-X support, who will quote you for a custom installation.

For administrator information about **MPI** for use with Vorpall, see [Running Vorpall from a Queueing System](#).

## Running Vorpall with mpiexec

In order to run Vorpall in parallel via the command line, you must first add the <VORPAL\_BIN\_DIR> to your PATH, as noted in the “Vorpall Command Line Options” section in the Reference Manual.

To run Vorpall in parallel, execute the following command:

```
mpiexec -n <#> vorpall -i filename.pre
```

in which <#> is the number of processors, **vorpall** is the executable program for parallel computations, and `filename.pre` is the name of the input file (which must be in the current directory, or must be specified by a full path).

Following **mpiexec**, but before **vorpall**, you can specify a variety of **mpiexec** options. In particular, for the open-mpi implementation of MPI (supplied with macOS), one may need to add the arguments, `-x PYTHONPATH -x LD_LIBRARY_PATH` to ensure that all processes are using the correct values for these environment variables. For the MPICH implementation of MPI (supplied with Linux) these arguments are not needed, as MPICH by default exports all environment variables to all processes. For more information about **mpiexec**, including the complete list of options, it can be run with **mpiexec -h**.

Following **vorpall**, you can specify a variety of Vorpall options. Some of the more common options are

```
vorpallser -i esPtclInCell.pre -o newesPtclInCell
```

```
vorpallser -i esPtclInCellSteps.pre -r 50
```

For a complete list of options, see the “Vorpall Command Line Options” section in the Reference Manual.

If a parameter is both set within the input file and specified on the command line, the command line parameter value takes precedence. The command line override enables you to configure an input file with default values while exploring alternative parameter settings from the command line. From the command line, you can quickly change simulation run lengths, dimensionality, output timing, etc.

Vorpall automatically adjusts its decomposition to match the number of processors it is given, unless a manual decomposition is provided for the correct number of cores in the input .pre file.

In contrast to Vorpall for serial computation, which creates a single text output file, Vorpall for parallel computation creates multiple text output files. Each individual processor from the parallel run sends comments to a different output file. A parallel computation output file’s name includes a label that identifies the number of the processor that generated that file, for example:

- esPtclInCell\_comms\_0\_1.txt
- esPtclInCell\_comms\_0\_2.txt

in which the final `_0` and `_1` before the file name suffix indicate the number of the processor.

By default Vorpall writes one HDF5 file for each field or particle species even for a parallel run. However, one can modify this behavior, as noted at the above link. Having one file for each field or particle species for each processor can sometimes get around parallel I/O problems. When that is necessary, one can construct a single file for a field or particle species using the utilities, `mergeH5Flds` and `mergeH5Ptcls`, which come with Vorpall.

## Running Vorpall with mpiexec Using a Hostfile

If you need to run an MPI job but do not have access to a queuing system then a hostfile must be set up. If this is the case you must know the node names on the cluster that the job is to be run on. You must then create a text file with your text editor of choice, this is your hostfile, and place it in your home directory. The hostfile simply contains each node name repeated on a new line as many times as there are threads in that node. For example consider a two node cluster with four threads each, the hostfile will contain

```
node1
node1
node1
node1
node2
node2
node2
node2
```

To run a job one must then source GSimComposer shell script using the command:

```
source <GSim_SCRIPT_DIR>/GSimComposer.sh
```

**Note:** This action changes your environment in your current shell, and so may make other programs fail. Do this in a separate shell from any shell in which you intend to run standard programs, like vi or emacs.

You are now ready to run in MPI using the *mpiexec* command with the above hostfile (signified as <hostfile> below. For MPICH (which is provided for Linux), the command is

```
mpiexec -f <hostfile> -n <#> <other mpiexec options> vorpall \
-i simulationname.pre <other vorpall options>
```

The equivalent command for openmpi (which is provided for macOS) is

```
mpiexec --hostfile <hostfile> -n <#> <other mpiexec options> vorpall \
-i simulationname.pre <other vorpall options>
```

The number of nodes, <#>, must be consistent with the computational resources and the hostfile.

## 10.2.2 Running Vorpall From a Queueing Systems

Running Vorpall with **MPI** either directly or with Parallel Queueing Systems requires use of different shell scripts to enable invocation of the Vorpall executable, as outlined below. In this section we discuss Linux queueing systems. For running Vorpall through Windows HPC Cluster Pack see *Running Vorpall on a Windows HPC Cluster*.

Queueing systems, such as PBS, LoadLeveler, LSF, SGE and Slurm, require the submission of a shell script with embedded comments that act as commands that the queueing system interprets. Below we show some of the more common embedded comments. Discussion of all the embedded comments is beyond the scope of this document. Furthermore, the command for submitting the job can vary. Below we will provide a common command, but you should contact the system administrator to ensure that you have the correct job submission command.

### Lustre File Systems

Many supercomputers use the Lustre file system, which has multiple Object Storage Targets (OSTs). Using more of these improves output speed for the large files produced by Vorpall in large-scale computing. The number to be used is set by a stripe command, such as

```
:
lfs setstripe -count C .
```

which sets the number of OSTs to C in the current directory. As noted at <https://www.nersc.gov/users/storage-and-file-systems/i-o-resources-for-scientific-applications/optimizing-io-performance-for-lustre/>, one should set C according to

File size	C
1-10GB	8
10-100GB	24
100GB+	72

## PBS/Torque/OpenPBS

Here is an example of a basic shell script for a PBS-based system.

```
#PBS -N vaclaunch
#PBS -l nodes=2:ppn=2
export GSim_DIR=$HOME/GSim-1.0
source $GSim_DIR/GSimComposer.sh
cd /directory/containing/your/input/file
mpiexec -np 4 vorpal -i vaclaunch.pre -n 250 -d 50
```

The *-l* commands relate to the resource requirements of the job. This file explicitly specified the number of cores per node, and number of nodes, so we have a total of four MPI ranks on which to execute the job, which is mirrored in the *mpiexec -np* argument.

If the contents of the above file are in *vaclaunch.pbs*, then the job would commonly be submitted by

```
qsub vaclaunch.pbs
```

although some MOAB based systems might use *msub*.

## Sun Grid Engine (SGE)

Here is another example, this time for a SGE (Sun Grid Engine) job.

```
## -cwd -V
## -l h_rt=0:10:00
## -l np=16
## -N magnetron2D
export GSim_DIR=$HOME/GSim-1.0
source $GSim_DIR/GSimComposer.sh
mpiexec -np 16 vorpal -i magnetron2D.pre
```

This time the *-cwd -V* tells the queue system to use the current working directory for the job, and to import the current environment and make this available for the script.

In this case, the queue system calculates the configuration based on the choice of 16 cores. On this cluster the *-l* commands is also used to specify the run duration, which, in this example, is set to 10 minutes.

If the contents of the above file are in *magnetron2D.qsub*, then the job would commonly be submitted by

```
qsub magnetron2D.pbs
```

## Platform LSF

Here is a third example, for the Platform LSF system:

```
#BSUB -o EBDP-VSim10.0.out
#BSUB -e EBDP-VSim10.0.err
#BSUB -R "span[ptile=16]"
#BSUB -n 32
#BSUB -J testVSim10.0
#BSUB -W 45
cd $HOME/electronBeamDrivenPlasma
export VSIM_DIR=$HOME/VSim-10.0
```

(continues on next page)



(continued from previous page)

```
source $VSIM_DIR/VSimComposer.sh

export MYJOB=electronBeamDrivenPlasma.pre

mpiexec -np 32 vorpal -i ${MYJOB}
```

With this submission system and job scheduler, `-W` is used to denote the wall time for the job in minutes. The name is passed by `-J`.

Sometimes one must reference a specific project for accounting purposes in the job submission file. For this you may use the `-A` option.

If the above commands are in the file, *electronBeamDrivenPlasma.lsf*, then the job is commonly submitted by

```
bsub < electronBeamDrivenPlasma.lsf
```

## Slurm

Here is an example from a Cray system with a custom Vorpal build running from a directory `/project/nnnnn/gnu-5.2.40`.

```
#!/bin/bash
#SBATCH --account=nnnnn
#SBATCH --job-name=lpa
#SBATCH --output=lpa.out
#SBATCH --error=lpa.err
#SBATCH --nodes=2
#SBATCH --time=00:05:00
srun --ntasks=32 --hint=nomultithread --ntasks-per-node=16 /project/nnnnn/gnu-5.2.40/
↳ vorpal-exported/bin/vorpal -i laserPlasmaAccel.pre -n 100 -d 20
```

If the file containing the above is named, *laserPlasmaAccel.slm*, then this job is submitted with

```
sbatch laserPlasmaAccel.slm
```

You can check on your job with

```
squeue -u $USER
```

and you can stop the job with

```
scancel JOBID
```

where JOBID is the job id returned by squeue.

### 10.2.3 Running Vorpai on a Windows HPC Cluster

**Note:** Prior to running GSim with Windows HPC Cluster tools, please ensure that GSim is properly installed on your Windows Cluster. (See the “Installation” Manual.)

#### Setting up the Simulation Directory

The following is an example of how to run an example GSim simulation on a Windows Cluster. In this example, the UNC Share Path that is set up on all nodes in the Cluster including the headnode is:

\\hpcheadnode\scratch

This path should be replaced by the path to your shared drive, whatever it might be. Paths in UNC (Universal Naming Convention) should be used, so \\machine\_name\name\_of\_share\directory instead of S:\directory.

To start, we create a new simulation in a directory on the shared drive. To do this, run GSimComposer on the headnode, which should be installed on the shared drive as *GSim must be installed in the shared drive to run Vorpai on a Windows Cluster*. shows.

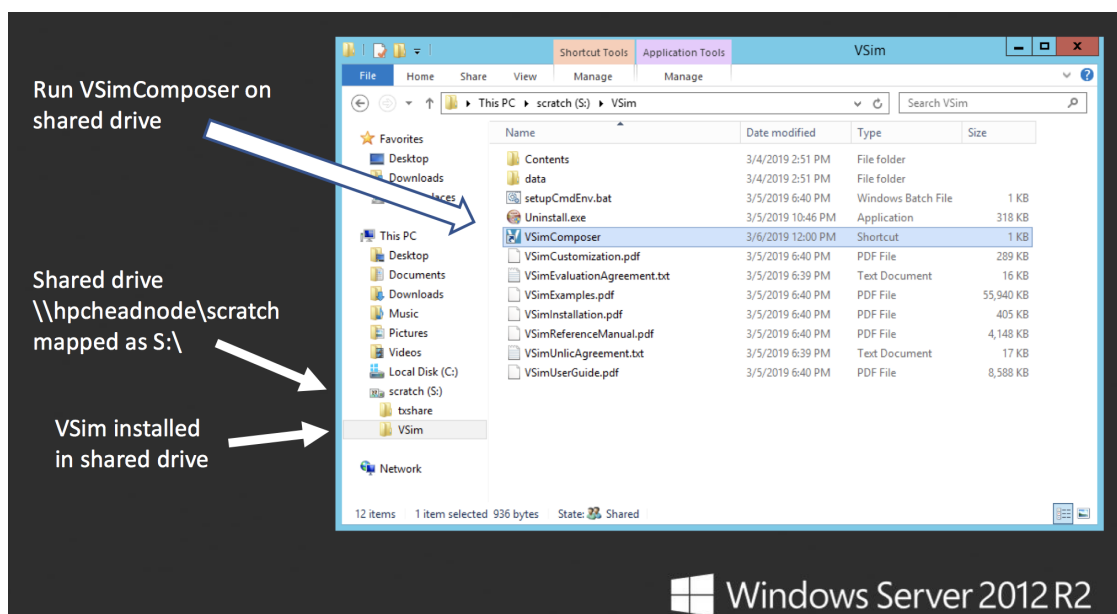


Fig. 10.12: GSim must be installed in the shared drive to run Vorpai on a Windows Cluster.

In GSimComposer, follow these actions:

- Select the *New → From Example...* menu item in the *File* menu.
- In the resulting *Examples* window expand the *GSimfor Basic Physics* option.
- Expand the *Basic Examples* option.
- Select *Parallel Plate Capacitor* and press the *Choose* button.
- Select the text in the *Directory* field and replace it with the shared drive UNC path. See *Choose the shared drive in the Directory field so that the subsequent New Folder is on that drive.* below.

Continue with:

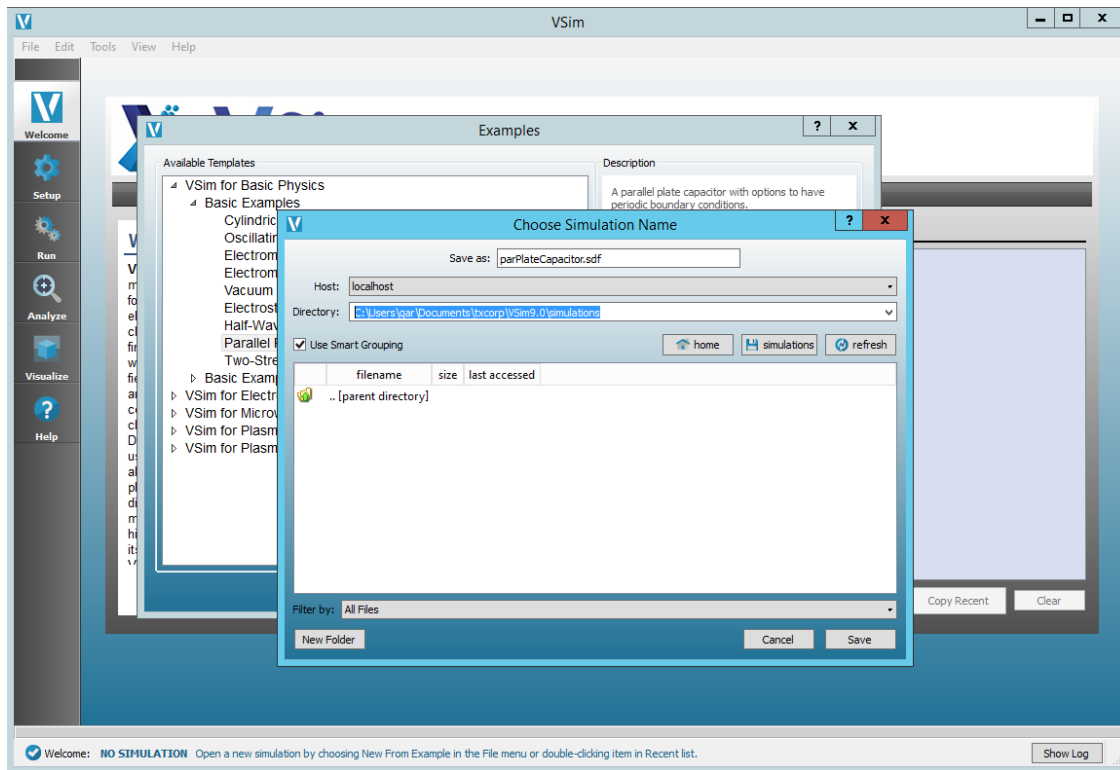


Fig. 10.13: Choose the shared drive in the Directory field so that the subsequent New Folder is on that drive.

- Select the *New Folder* button and type in *parPlateCapacitor* for the name. See *Choose the name of the containing directory of your new simulation.* below.

Now:

- Click the *Create* button in the *New Folder* dialog.
- Click the *Save* button in the *Choose Simulation Name* dialog.
- At this point, you would normally change the setup to suit your simulation needs and possibly run it locally to check if you are on the right track, but for now, just exit GSimComposer.

You should now have a simulation directory in your share drive and be able to see it in Windows Explorer. See *The simulation directory needs to be on the shared drive to run a simulation on a Windows Cluster.*

## Create a New Cluster Job

Jobs can be started from the command line and from the Cluster Manager tool. We will show how to create and submit a job from the Cluster Manager interface and leave it to the user to follow the Microsoft documentation on how to save the Job XML and run subsequent jobs from the command line.

Start the Cluster Manager tool usually located in

C:\Program Files\Microsoft HPC Pack 2012\Bin\HpcClusterManager.exe

This should present with a window as shown in *The Microsoft HPC Pack Cluster Manager tool used to create and submit jobs to the cluster.*

In the Cluster Manager, complete the following steps:

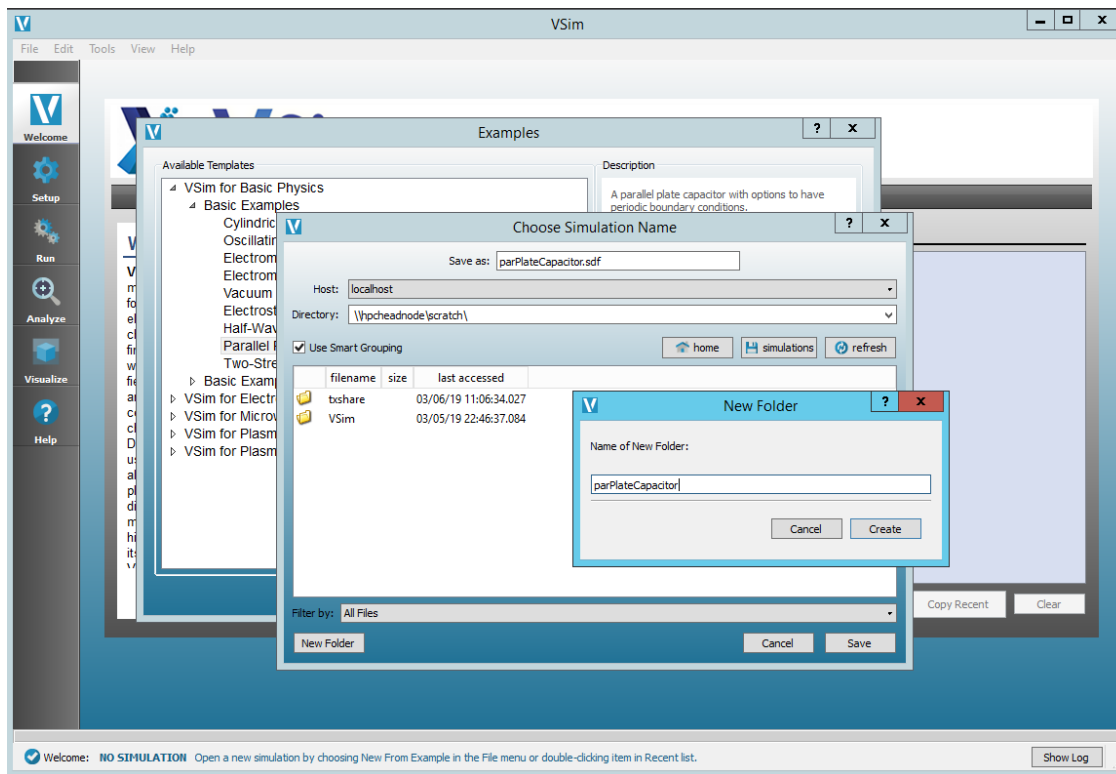


Fig. 10.14: Choose the name of the containing directory of your new simulation.

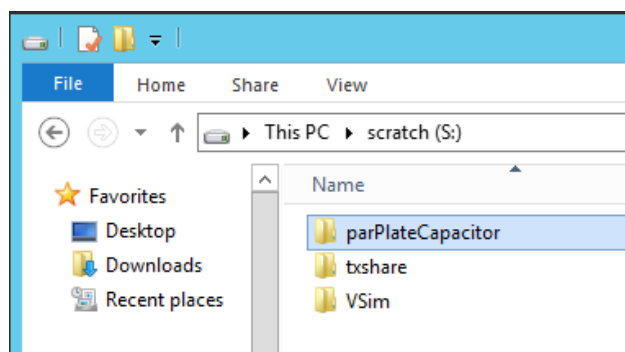


Fig. 10.15: The simulation directory needs to be on the shared drive to run a simulation on a Windows Cluster.

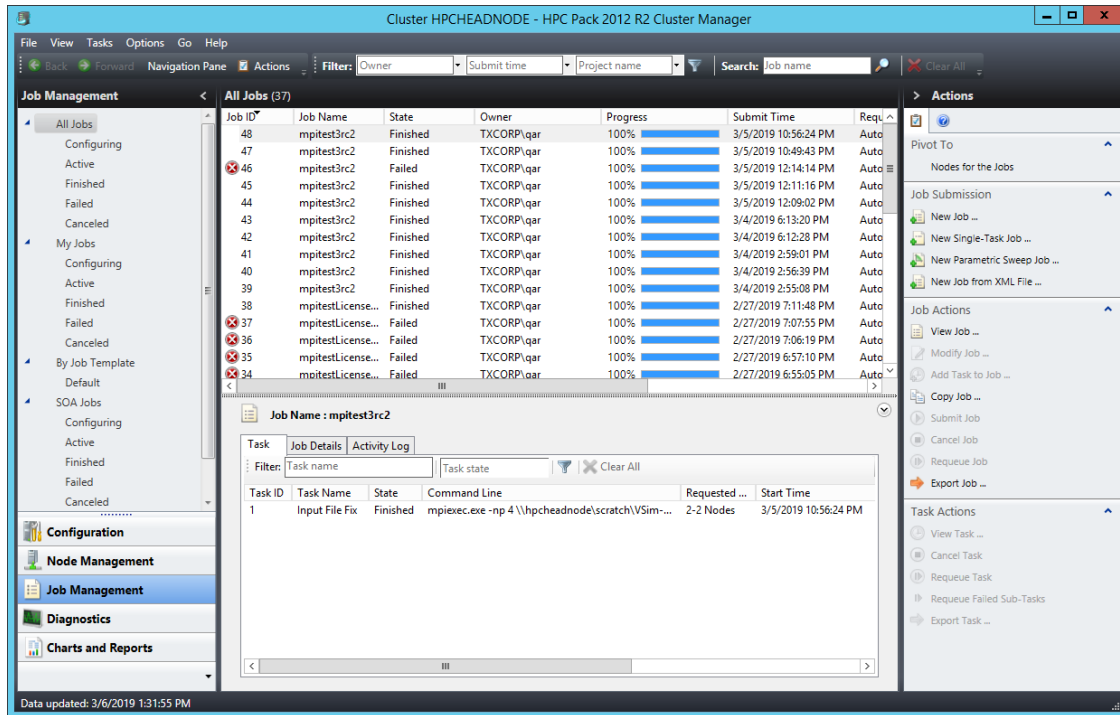


Fig. 10.16: The Microsoft HPC Pack Cluster Manager tool used to create and submit jobs to the cluster.

- Select the *Job Management* tab in the lower left part of the window.
- In the right *Actions* pane click *New job...* This will bring up a New Job dialog
- In the *New Job* dialog, type a name in the *Job name* field as shown in *The Cluster Manager New Job dialog is a wizard to help create and submit a job.*

Continue with:

- Select the *Edit Tasks* item in the left menu.
- Click the Add button as shown in *Adding a task in the Cluster Manager New Job dialog.*

Edit the task details by:

- Edit the *Task name* field if desired.
- Enter the command for running vorpall via MPI into the *Command line* field. Here you need the full path. to the vorpall executable and input file for your simulation. All the normal vorpall command-line arguments apply.
- The *Working directory* field is required to be the simulation directory that we set up above.
- The *Standard output* and *Standard error* fields are optional, but make it handy to organize the output.
- Finally, select the number of nodes you would like to run on. The “-np 4” argument along with the Minimum 2 value says that we would like to run 4 MPI processes across 2 nodes.

The job Task Details dialog is shown in *Creating parameters for the task in the Cluster Manager New Job dialog.*

Once the job task details are set, we need to do one last set of steps:

- Select the task defined above.
- Click *Environment Variables* in the *Task Properties* list. Then click the ... button to the right of the line. The is highlighted in *Editing Job Properties for the task in the Cluster Manager New Job dialog..*

**New Job**

**Job Details**

Job name:

Job template:

Project:

Priority:

**Job run options**

☐ Do not run this job for more than:

Days:  Hours:  Minutes:  Seconds:

☐ Run job until cancelled or run time expires

☐ Fail the job if any task in the job fails

Send a notification when this job:

☐ Starts ☐ Completes

Send email notifications to:

Email, or other notifications must be previously configured by a cluster administrator.

**Job resources**

Select the type of resource to request for this job:

Enter the minimum and/or maximum of the selected resource type that this job is allowed to use:

Minimum: ☐ Auto-calculate ☐ 1

Maximum: ☐ Auto-calculate ☐ 1

☐ Use assigned resources exclusively for this job

No other jobs will be allowed to run on the selected nodes while the job is running.

☐ Run the entire job on a single node

Fig. 10.17: The Cluster Manager New Job dialog is a wizard to help create and submit a job.

**New Job**

**Job Details**

**Edit Tasks**

Resource Selection

Licenses

Environment Variables

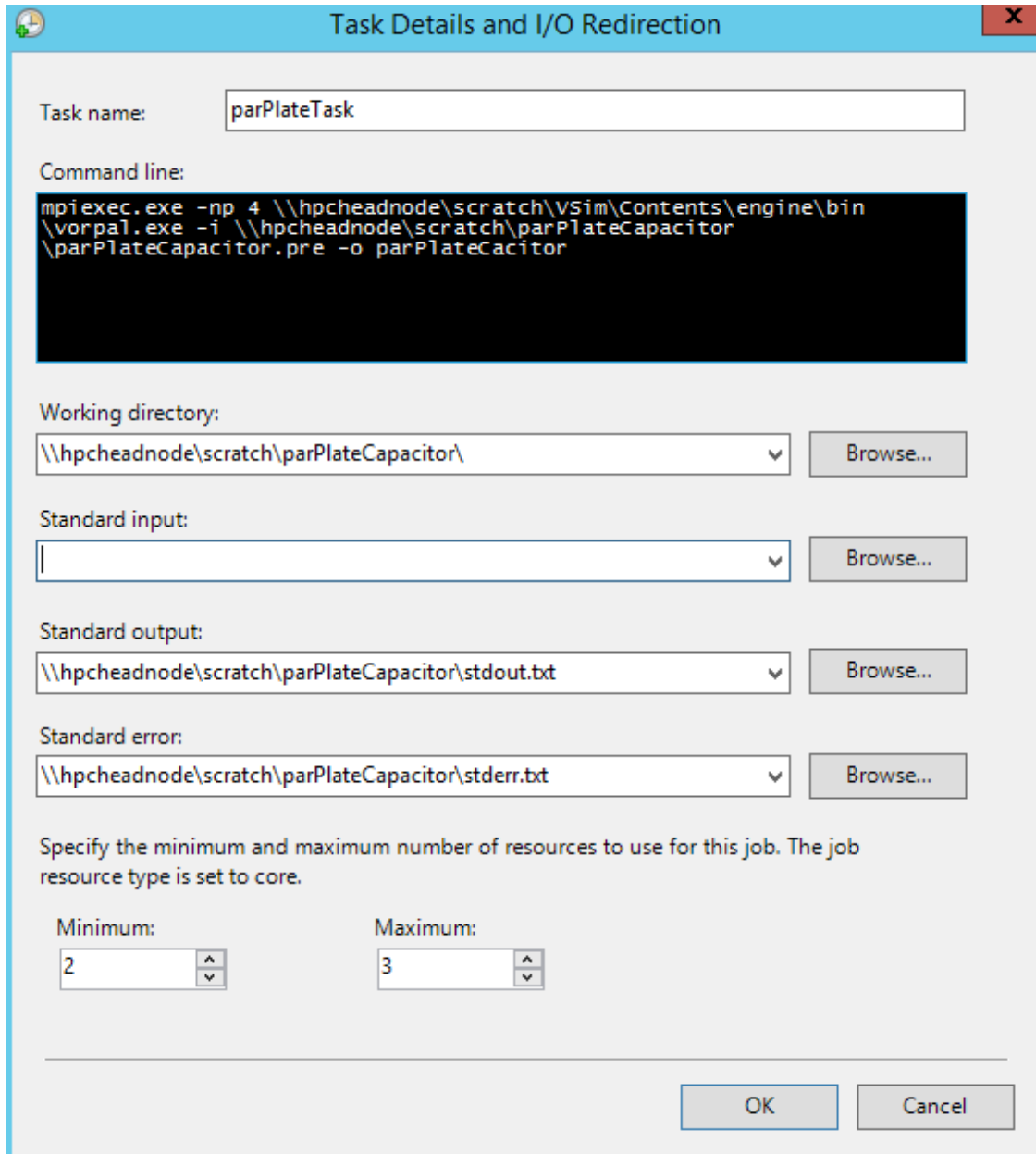
Advanced

Specify tasks for this job. [More about tasks and task types](#)

**Tasks**

Task Na...	Type	Command Line	Requested Resources

Fig. 10.18: Adding a task in the Cluster Manager New Job dialog.



The dialog box is titled "Task Details and I/O Redirection". It contains the following fields and controls:

- Task name:** A text box containing "parPlateTask".
- Command line:** A text area containing the command:
 

```
mpiexec.exe -np 4 \\hpcheadnode\scratch\VSIm\Contents\engine\bin
\vorpai.exe -i \\hpcheadnode\scratch\parPlateCapacitor
\parPlateCapacitor.pre -o parPlateCacitor
```
- Working directory:** A dropdown menu showing "\\hpcheadnode\scratch\parPlateCapacitor\" with a "Browse..." button to its right.
- Standard input:** A dropdown menu with an empty field and a "Browse..." button to its right.
- Standard output:** A dropdown menu showing "\\hpcheadnode\scratch\parPlateCapacitor\stdout.txt" with a "Browse..." button to its right.
- Standard error:** A dropdown menu showing "\\hpcheadnode\scratch\parPlateCapacitor\stderr.txt" with a "Browse..." button to its right.
- Resource specification:** A section with the text "Specify the minimum and maximum number of resources to use for this job. The job resource type is set to core." Below this are two spin boxes:
  - Minimum:** Set to 2.
  - Maximum:** Set to 3.
- Buttons:** "OK" and "Cancel" buttons at the bottom right.

Fig. 10.19: Creating parameters for the task in the Cluster Manager New Job dialog.

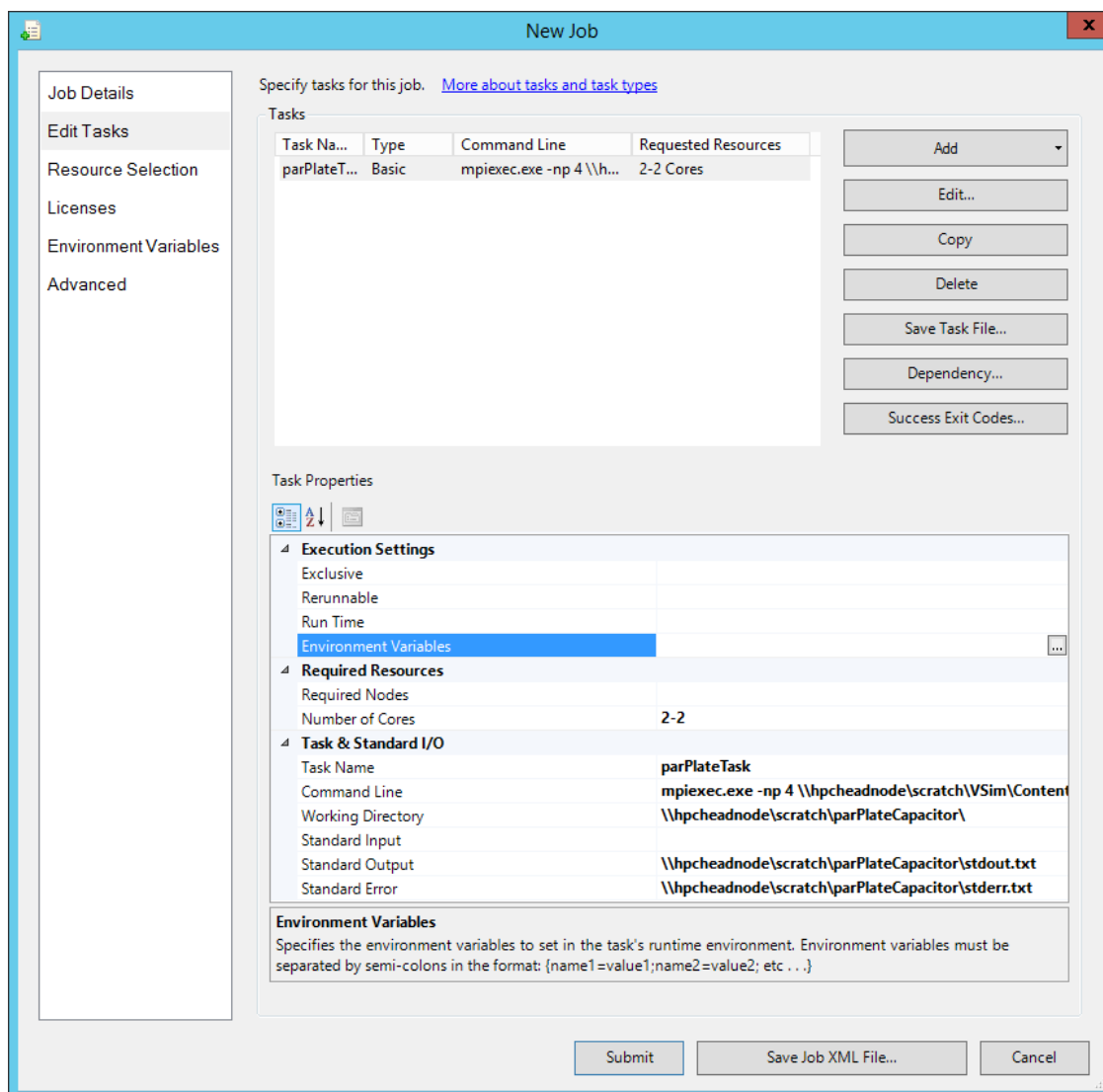


Fig. 10.20: Editing Job Properties for the task in the Cluster Manager New Job dialog.



In the Environment Variables dialog (*Environment Variables dialog for the task allows user to see and add variables.*),

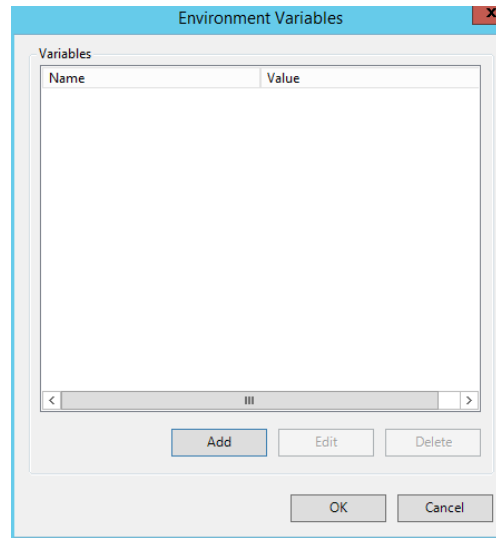


Fig. 10.21: Environment Variables dialog for the task allows user to see and add variables.

click the *Add* button. This will bring up the *Add Environment Variable* dialog (see *Add Environment Variables dialog for the task to add PYTHONPATH.*).

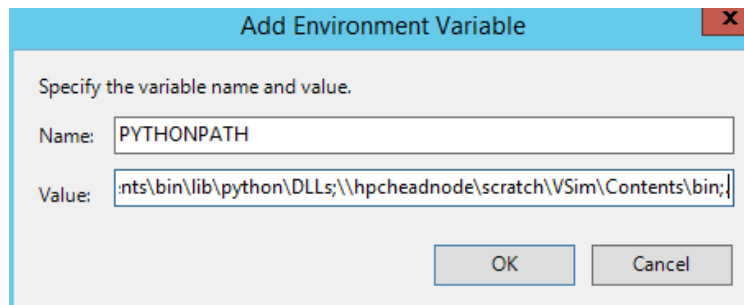


Fig. 10.22: Add Environment Variables dialog for the task to add PYTHONPATH.

In this dialog, enter PYTHONPATH for the *Name* and the following for *Value*:

```
\\hpheadnode\scratch\GSim\Contents\engine\share\scripts;\\hpheadnode\scratch\GSim\
→ Contents\engine\lib\site-packages;\\hpheadnode\scratch\GSim\Contents\engine\lib;\\
→ hpheadnode\scratch\GSim\Contents\bin\lib\site-packages;\\hpheadnode\scratch\GSim\
→ Contents\bin\lib\python\lib;\\hpheadnode\scratch\GSim\Contents\bin\lib\python\DLLs;\\
→ hpheadnode\scratch\GSim\Contents\bin;.
```

You will want to replace the value of the UNC Share Path (\\hpheadnode\scratch) with your own UNC Share Path in the PYTHONPATH line. The PYTHONPATH variable is required for all vorpal simulations to run, but other variables such as SIM\_DATA\_PATH may be needed for a few select simulations.



## OUTPUT DATA

### 11.1 HDF5 Format Data Output Files

Vorpal outputs data in two forms, text and HDF5. Text output is used for progress reporting, while HDF5 is used for data files. The HDF5 data files have the `.h5` suffix.

Hierarchical Data Format Version 5 (HDF5) is a library and file format for storing graphical and numerical data and for transferring that data between computers. Vorpal and VSimComposer output data in HDF5 format. The Hierarchical Data Format was developed by the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign. For more information about HDF5 (See <http://hdfgroup.org/HDF5>).

### 11.2 Frequency for dumping simulation quantities

The following are the general dumping options that can be set in field, particle, and grid boundary blocks. These can be used when the user wishes to customize the dumping options for the particular blocks and not change the global dumping options.

- `dumpOncePerRun` (boolean)

If true, this object will be dumped only once per run. This specification can co-exist with other specifications.

- `commandLineDumpPeriodicity` (integer)

This option may be automatically set by vorpal to override other attributes specifying when this object should dump its data.

- `dumpPeriodicity` (integer)

If `dumpPeriodicity = p`, then this object will be dumped to disc when its timestep mod `p` equals 0.

If `p=0`, this object “will never be dumped.

- `dumpPeriod` (integer)

(deprecated: please use `dumpPeriodicity` instead)

If `dumpPeriodicity = p`, then this object will be dumped to disc when its timestep mod `p` equals 0.

If `p=0`, this object will never be dumped.

- `dumpSteps` (integer list)

A list of the time steps at which this object should dump its data to disc.

- `dumpSteps` (expression)

A function of a scalar integer  $n$  (the current time step) that returns 1 if this object should be dumped at step  $n$ , and 0 if not.

To use an Expression, it must be specified within the <Expression dumpSteps> block:

```
expression = (a function of 'n', the timestep, that yields 0 or 1)
For example, expression = ( mod(n, 100) == 0 )
or          expression = ( or(n==100, n==200) )
```

Vorpal produces one HDF5 file for each field or species at each dump time. For example, if the simulation parameter `nsteps = 100` and the simulation parameter `dumpPeriodicity = 10`, Vorpal dumps data 10 times during the simulation and outputs a total of 10 HDF5 files for each field or species while running the simulation. Vorpal also produces one `Globals` file at each dump containing data that is general to the whole simulation, as opposed to one species or field. Finally, Vorpal puts out a `History` file containing the data of the specified histories.

## 11.3 Change the Names of Output Files

If you want to change the names of the output files, which include the `.h5` files, you can specify the `-o` output option when you run Vorpal.

For example, you want to replace `emPlaneWave` with `emPlaneWaveTest1` in the names of the `emPlaneWave` simulation's output files. Run Vorpal from the command line using this command:

```
vorpal -i emPlaneWave.in -o emPlaneWaveTest1
```

Then the text output files would be:

- `emPlaneWaveTest1_all_1.txt`
- `emPlaneWaveTest1_comms_0.txt`
- `emPlaneWaveTest1_completed.txt`
- `emPlaneWaveTest1_dumpedobjs_0.txt`

and a field output files might be:

- `emPlaneWaveTest1_edgeE_1.h5`

## 11.4 Displaying the Content of .h5 Files

The **h5dump** utility converts the binary data in `.h5` files into human-readable ASCII data in `.txt` files, and is available for all the platforms on which Vorpal runs. You can download the utility from the HDF5 website (<http://hdfgroup.org/HDF5>).

The basic command is:

```
h5dump -o output_file_name.txt your_h5_file.h5
```

So, to convert the `emPlaneWave_edgeE_1.h5` to text format, use this command:

```
h5dump -o emPlaneWave_edgeE_1.txt emPlaneWave_edgeE_1.h5
```

The **HDFView** is a GUI code for displaying the contents of `hdf5` files. Example output is shown below. HDFView may be downloaded for free from the HDF Group (<http://www.hdfgroup.org/hdf-java-html/hdfview/>)

## 11.5 Structure of Simulation Output .h5 Files

For each type of output file below, main data entries within that output file are displayed as a list of fields at the same level within the list. For those data fields within an output file that contain one or more subcategories of data, subcategories appear in an indented list below the main data category to which the subcategories apply.

### 11.5.1 Globals file

```
compGridGlobal
runInfo
time
```

### 11.5.2 GridBoundary file

```
name
  int array [NX NY NZ 2]
  name[0] = true (1) or false (0) is the lower left front corner inside or not?
  name[1] = true (1) or false (0) is the cell center inside or not?
nameLargeBndryFaces
nameLargeFaceFracs
nameSmallBndryFaces
nameSmallFaceFracs
nameStairStepBndryEdgesData
nameStairStepBndryFacesData
compGridGlobal
compGridGlobalLimits
derivedVariables
poly
runInfo
time
```

### 11.5.3 SumRhoJ file

```
SumRhoJ
  int array [NX NY NZ 4]
  SumRhoJ[0] = Rho, charge density
  SumRhoJ[1] = Jx
  SumRhoJ[2] = Jy
  SumRhoJ[3] = Jz
compGridGlobal
compGridGlobalLimits
runInfo
time
```

## 11.5.4 emMultiField field (MagMultiField) file

MagMultiField

```
int array [NX NY NZ 3]
MagMultiField[0] = Bx
MagMultiField[1] = By
MagMultiField[2] = Bz
```

compGridGlobal

compGridGlobalLimits

derivedVariables

runInfo

time

Below is an hdf5view of a field file.

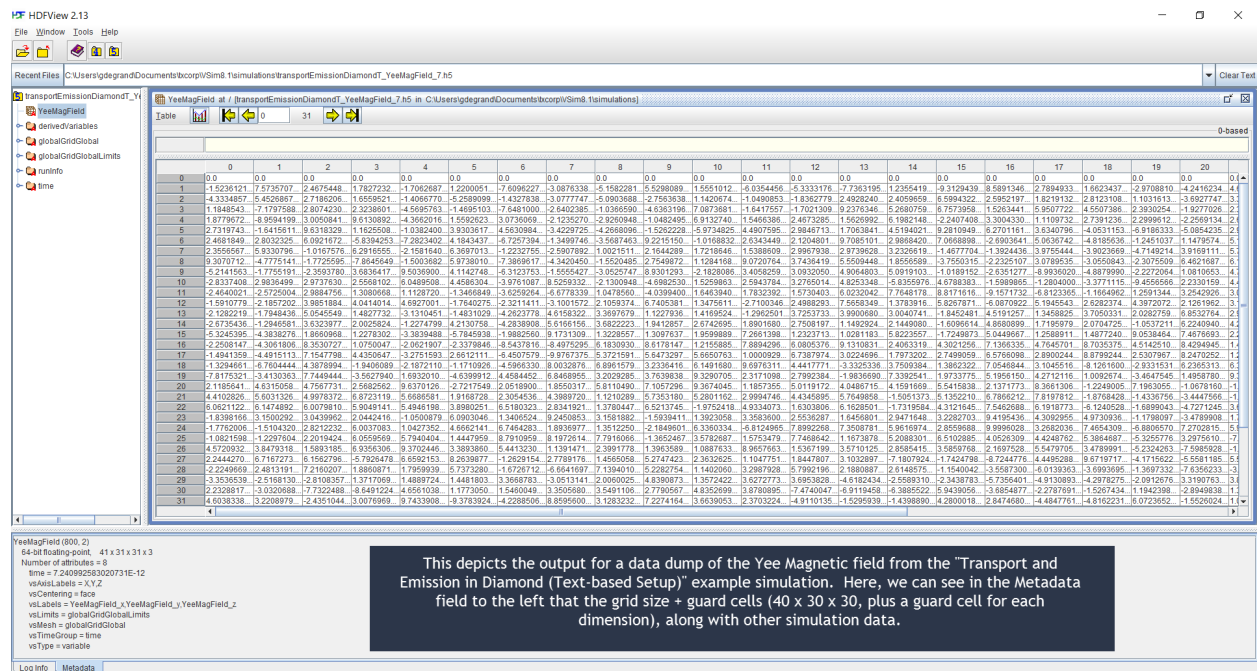


Fig. 11.1: HDFView display of a field file.

## 11.5.5 History file

runInfo

historyName1

historyName2

**Note:** These historyName arrays contain time data pertaining to the type of history chosen in the input file.

### 11.5.6 Fluid file

```
NeutralGasName
  int array [NX NY NZ 1]
  NeutralGasName[0] = Density
compGridGlobal
compGridGlobalLimits
runInfo
time
```

### 11.5.7 Species (particle) file

```
species
  int array [NX NY NZ 6]
  species[0] = x position
  species[1] = y position
  species[2] = z position
  species[3] = x velocity
  species[4] = y velocity
  species[5] = z velocity
compGridGlobalLimits
runInfo
time
```

**Note:** The information above is representative of 3D data. The actual number of elements in an array may vary depending on the dimensionality of the simulation. The number of elements in a species output file will also vary based on the type of species used. For more information on the output of species data, see the next section.

Below is an hdfview display of a species (particle) file

### 11.5.8 Columns in Species (Particle) .h5 Output Files

Below is a table displaying how the columns in particle simulation .h5 output files for various species kinds correspond to the columns that can be seen in the .h5 file when the file is opened using a tool such as **HDFView**. HDFView may be downloaded for free from the HDF Group at <http://www.hdfgroup.org/hdf-java-html/hdfview/>. HDFView distributions are available for 32-bit and 64-bit Linux, Mac, and Windows platforms.

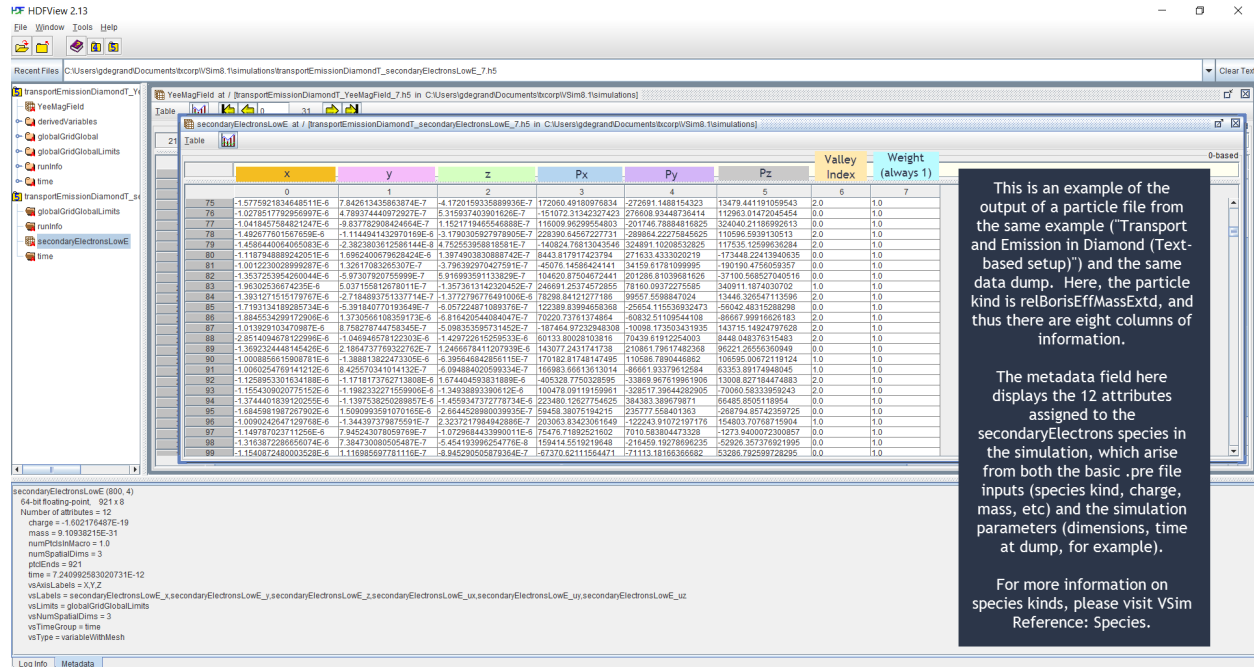


Fig. 11.2: HDFView display of particle file.

Table 11.1: Columns in .h5 in Particle Simulation Output Files

Species	Number of Columns	Comma-separated Columns
cmplxRelBorisDF	5 + NDIM	x,[y,[z]]Px, Py, Pz, real weight, imaginary weight
envBoris	5 + NDIM	x,[y,[z]]Px, Py, Pz, tag, weight
freeRel	3 + NDIM	x,[y,[z]]Px, Py, Pz
freeRelVW	4 + NDIM	x,[y,[z]]Px, Py, Pz, weight
noMove	3 + NDIM	x,[y,[z]]Px, Py, Pz
noMoveVW	4 + NDIM	x,[y,[z]]Px, Py, Pz, weight
nonRelBoris	3 + NDIM	x,[y,[z]]Px, Py, Pz
nonRelEs	3 + NDIM	x,[y,[z]]Px, Py, Pz
relBoris	3 + NDIM	x,[y,[z]]Px, Py, Pz
relBorisBallisticVW	4 + NDIM	x,[y,[z]]Px, Py, Pz, weight
relBorisCyl	3 + NDIM	u0,[u1,[u2,]]P0, P1, P2 (see Legend)
relBorisDF	4 + NDIM	x,[y,[z]]Px, Py, Pz, weight
relBorisEffMassExtnd	2 + 3(NDIM always 3)	x, y, z, Px, Py, Pz, valley index, weight (always 1)
relBorisFuncVW	4 + NDIM	x,[y,[z]]Px, Py, Pz, weight
relBorisTagged	4 + NDIM	x,[y,[z]]Px, Py, Pz, tag
relBorisVW	4 + NDIM	x,[y,[z]]Px, Py, Pz, weight
relBorisVWTagged	5 + NDIM	x,[y,[z]]Px, Py, Pz, tag, weight
relBorisVWScale	6 + NDIM	x,[y,[z]]Px, Py, Pz, tag, scale parameter, weight



Table 11.2: **Legend**

NDIM	Number of Dimensions: 1, 2, or 3	
	Dimension Notation	
1D	2D	3D
x	x y	x y z
1D	1D,[2D]	1D,[2D,[3D]]
	Cylindrical Coordinates	
Polar	u0, u1, u2	r, phi, z
Cylindrical	u0, u1, u2	z, r, phi
Tubular	u0, u1, u2	phi, z, r
	Momentum/Mass Notation Convention	
	momentum/mass = $\gamma v = P$	
$\gamma v_x: P_x$	$\gamma v_y: P_y$	$\gamma v_z: P_z$



## DESIGN TAB (PARAMETER SCANS)

### 12.1 Introduction to Parameter Scan

It is possible to run Parameter Scan (also known as Variable Sweep) through VSimComposer. You will be able to choose variable(s) and different values for them so that multiple simulations are run with the chosen values. In VSimComposer, each such simulation is called a **sub simulation**.

For the scan, You will be able to choose variables defined in the *Elements Tree* in visual setup and *Editor Pane* in the text-based setup. All *User Defined Constants* and *Parameters* that are not expressions (i.e. a constant) are eligible for the scan.

In the upcoming section, we will see how to run a parameter scan in VSimComposer. We will use an example **Dipole Above Conducting Plane** for the demonstration.

### 12.2 Using An Example To Demonstrate How To Run Parameter Scan

First open the **Dipole Above Conducting Plane** and follow along:

- Select the *New -> From Example* menu item in the *File* menu.
- In the resulting *Examples* window, expand the *VSim for Electromagnetics* option.
- Expand the *Antennas* option.
- Select *Dipole Above Conducting Plane* and press the *Choose* button.
- In the resulting dialog, create a new folder if desired, and press the *Save* button to create a copy of this example in your run area.
- This will open the **Dipole Above Conducting Plane** example.

Once the simulation is open and setup is completed, parameter scan can be performed by clicking on the **Design** icon from the icon panel (see [Fig. 12.1](#)).

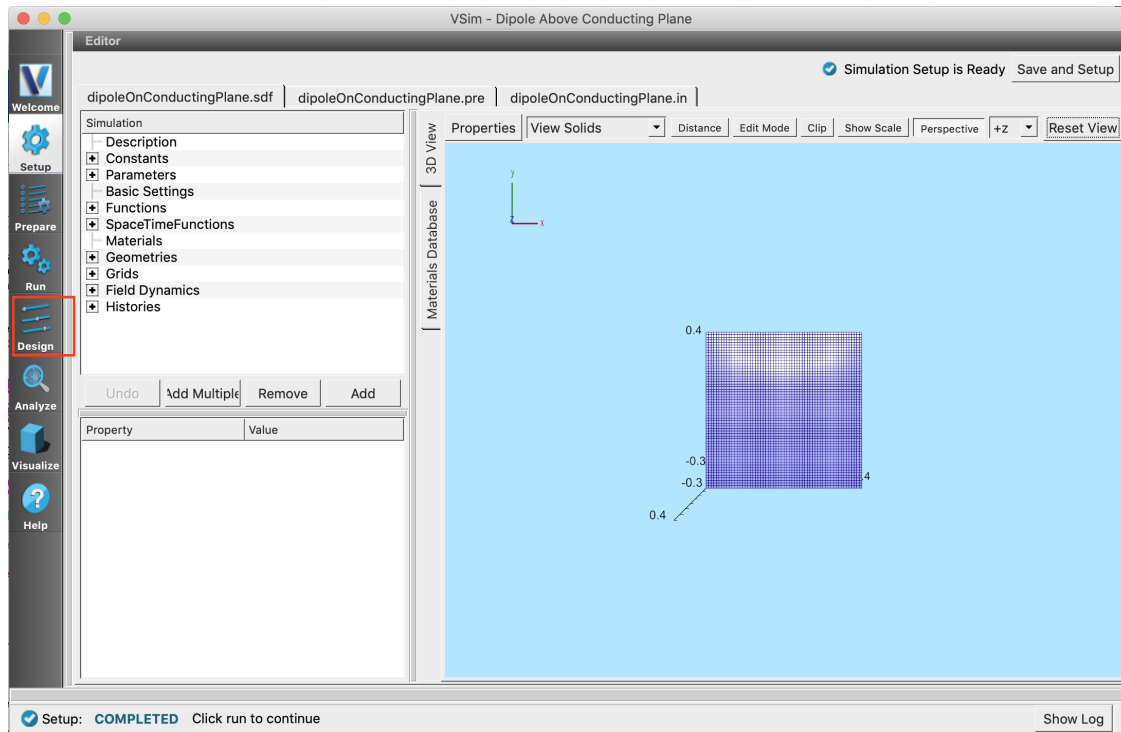


Fig. 12.1: Setup Window for Dipole Above Conducting Plane Example and Design Icon

## 12.3 Parameter Scan Panel

On clicking on the **Design** icon, the following **Parameter Scan** panel is visible.

## 12.4 Setting up the Parameter Scan

The *Runtime Options* pane allows setting up the scan and the *Logs and Outputs Files* pane allows viewing the status and logs of a **sub simulation** run.

- In *Runtime Options* pane, under the *Setup* tab, you will be able to set the name of the parameter scan. By default, the scan is named *Scan0* but for this exercise let's set it to *testScan*.
- The *Choose a Variable* drop-down lets us add the variables for the scan. VSIMComposer also supports scanning multiple variables at once. The chosen variables get added to the list below the drop down.
- The variable specific options for each added variable in the list are available upon the selection and can be set individually. See [Choosing Model And Values for the Variable](#).
- For this example, Choose the variable **HEIGHT**.

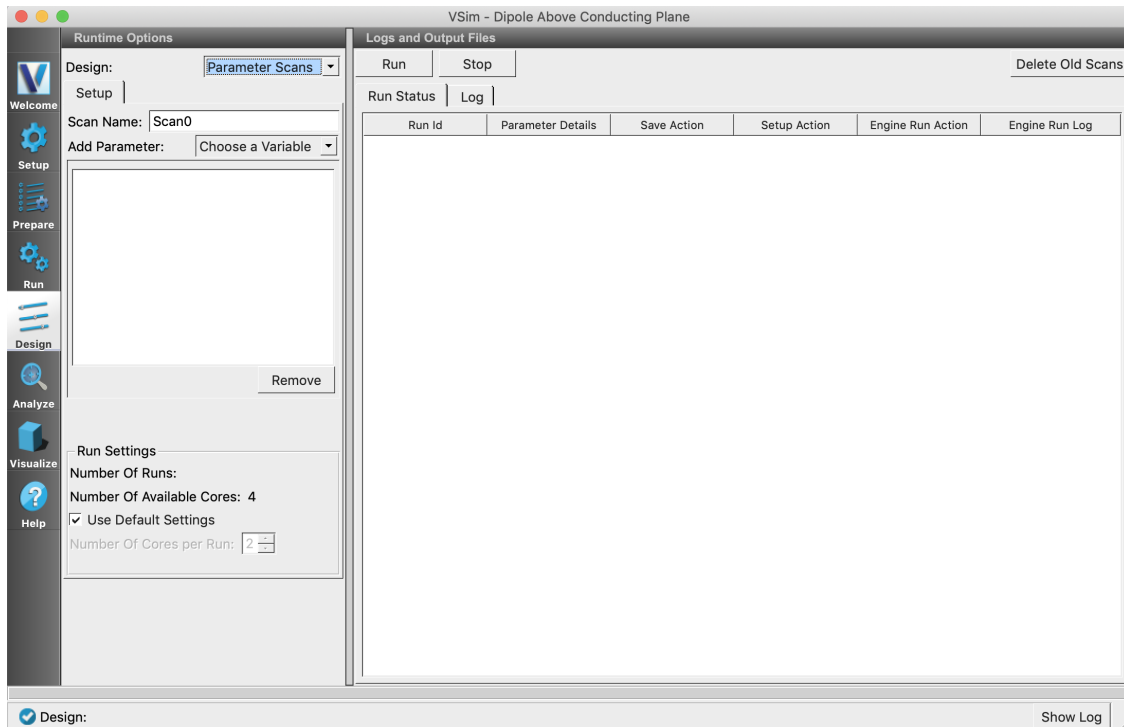


Fig. 12.2: Parameter Scan Panel

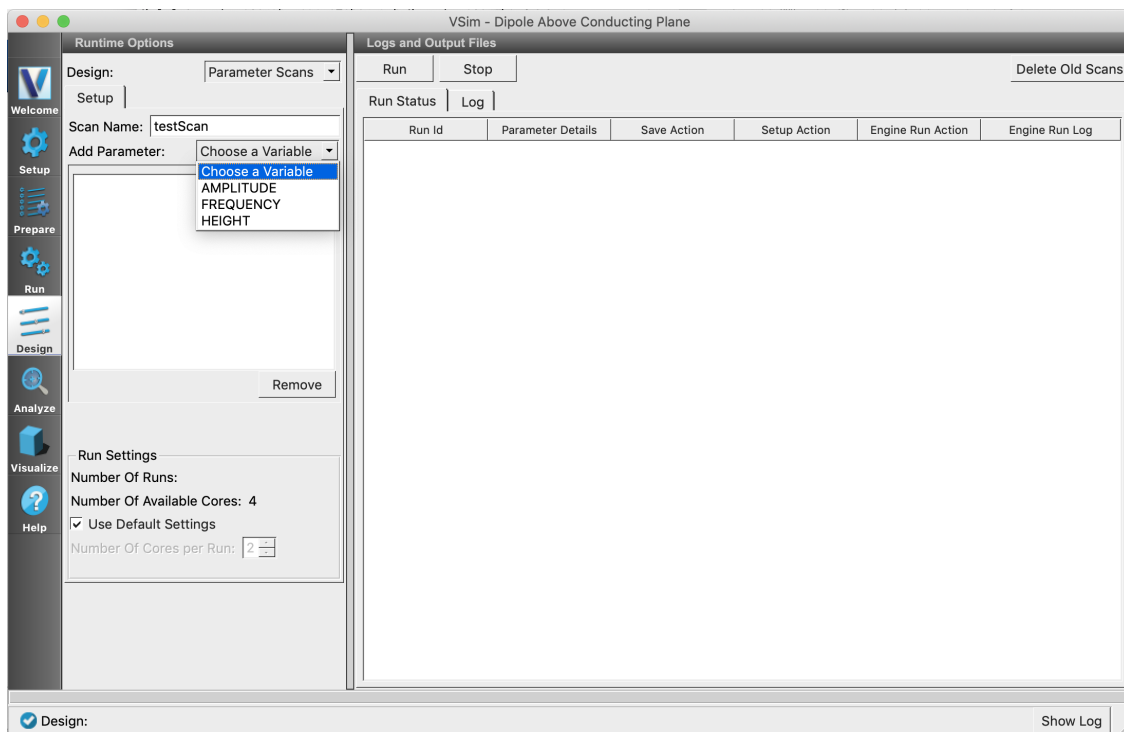


Fig. 12.3: Variable List

## 12.4.1 Choosing Model And Values for the Variable

The options below are available for each selected variable on the list.

- **Model Choice** Allows selection of the *Model*. *Model* refers to the way a variable gets different values for the scan.
  - *Range* : This model calculates values within the specified range (inclusive). It has *Number of Points* (total number of values), *Minimum* (start value) and *Maximum* (end value) fields. Checking the *Log Scale* steps the variable in log scale. By Default, number of points is 1, minimum is the current value of the variable and maximum is a number greater than the minimum by an arbitrary amount.

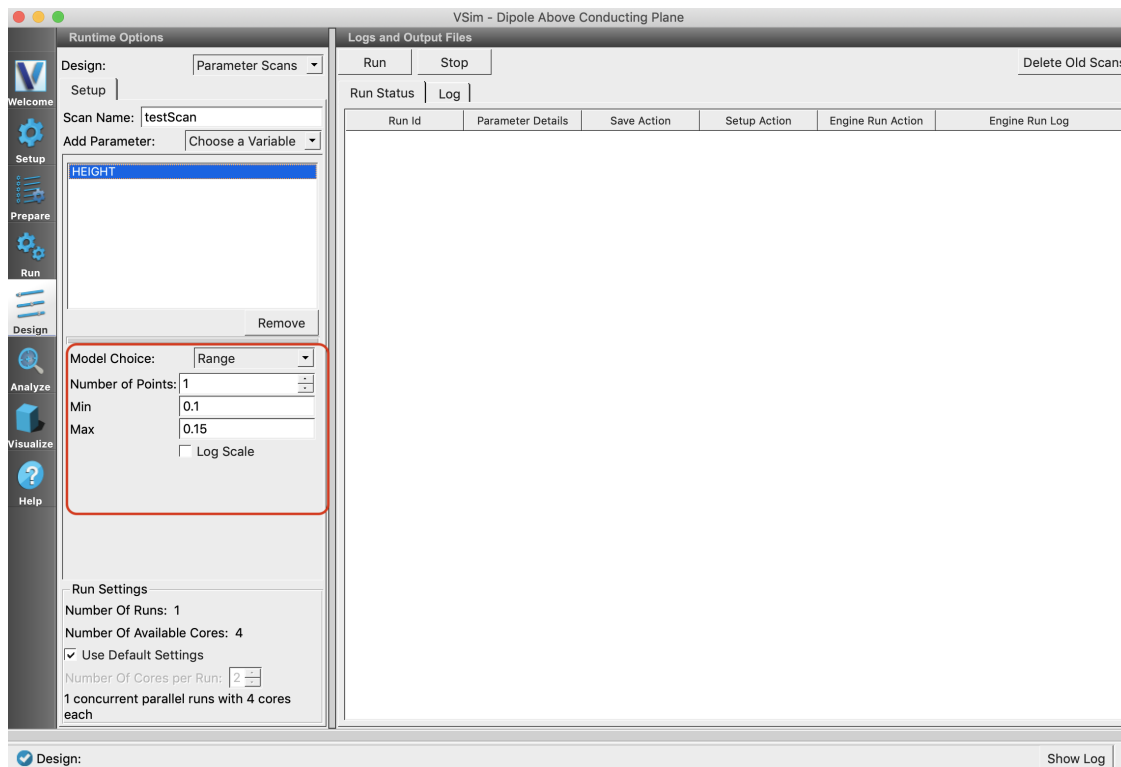


Fig. 12.4: Range Model

- *List* : Assigns specified discrete values to the variables. There are no default values for list.
- For this simulation Choose Range as the model choice, set number of points to 3, minimum to 0.15 and maximum to 0.3. There will be three sub simulations.

**Note:** For Range model, the values can only be incremented so always make sure the start value is less than the end value.

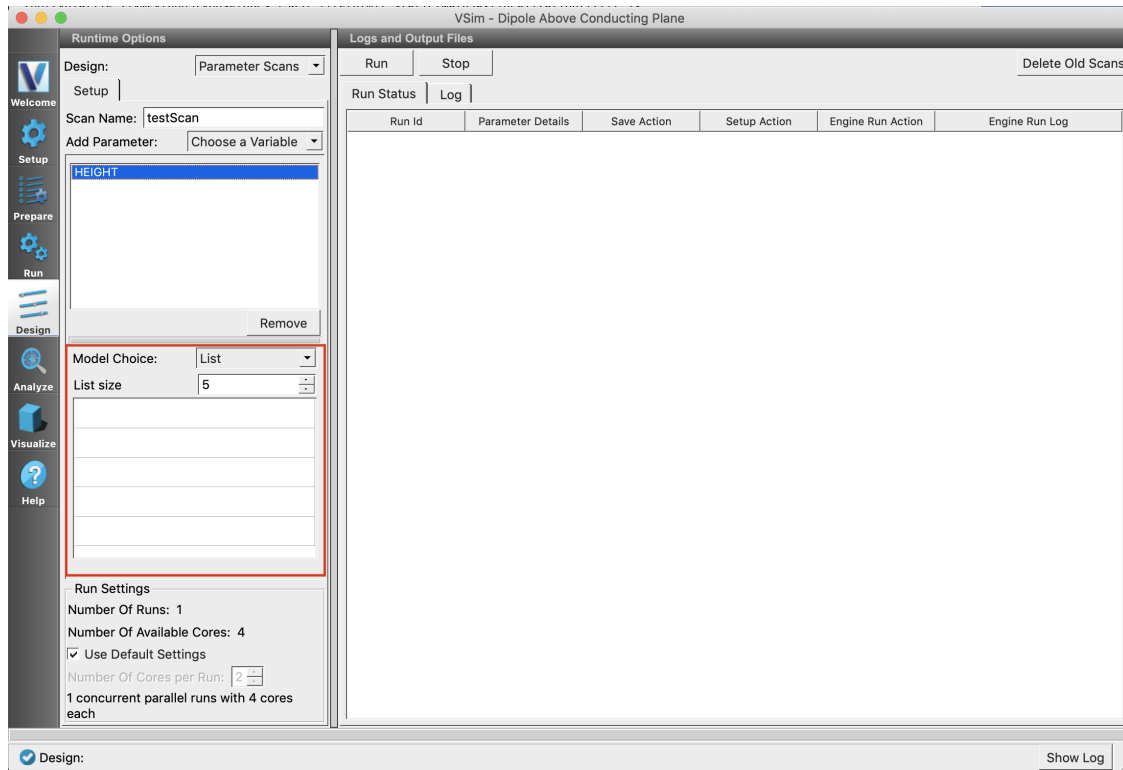


Fig. 12.5: List Model

### 12.4.2 Run Settings

The engine run parameters (Time Step, Dump Periodicity, Number of Steps) for each **sub simulation** are extracted from the **Run Window**. First we will define a few terms here and then see how the *Run Settings* option can be used.

- Sequential Runs: Run each **sub simulation** sequentially, i.e. out of  $n$  **sub simulations** only one will run at a time. Second run starts only after completion of the first.
- Concurrent Runs: **Sub simulations** are run concurrently, i.e.  $m(\leq n)$  number of **sub simulations** will run at a time.
- Parallel Run: Run each **sub simulation** with `mpirun` with certain `-np` value.
- Serial Run: Run each **sub simulation** in serial.

By default, *Use Default Settings* is selected and VSimComposer tries to maximize the number of licensed cores utilization by running as many **sub simulations** as possible. Each run can be serial or parallel depending on total number of runs and number of available licensed cores.

Unchecking the *Use Default Settings* ignores the default behaviour and allows you to enter the number of cores you want to run each **sub simulation** with. The value can be entered in *Number Of Cores per Run* box and cannot exceed the total number of licensed cores. To maintain consistency, all the **sub simulations** will run with the same number of cores.

VSimComposer informs about the type of run through a text label below the *Number Of Cores per Run* box.

For our **Dipole Above Conducting Plane** example, we will proceed with the default.

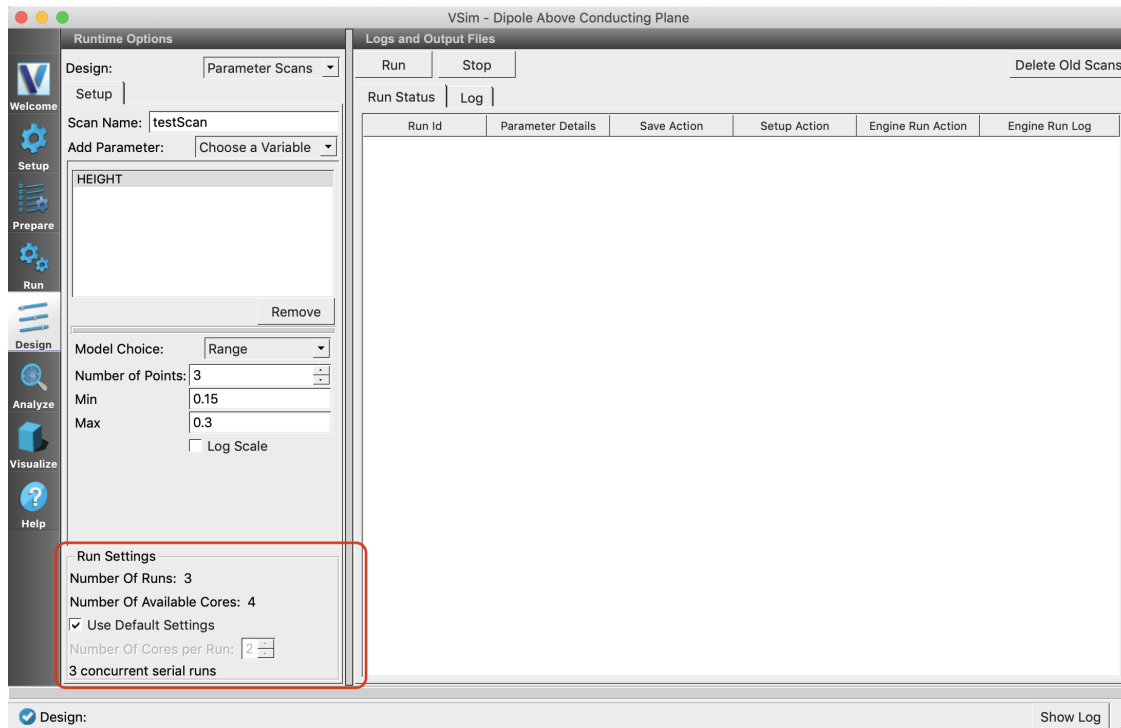


Fig. 12.6: Run Settings

## 12.5 Start/Stop the Scan

Click **Run** button to start the parameter scan. It can be stopped any time by clicking on **Stop**. On clicking **Stop** all the ongoing **sub simulation** runs are stopped and waiting runs will not be started.

## 12.6 Output pane

- **Run Status Tab** : This tab consists of a table that tracks the status of every step of each **sub simulation**. See Fig. 12.8. There are seven columns by default.
  - *Run Id* : Identification string for a sub simulation. Contains name of the scan and index of variables.
  - *Parameter Details* : Contains a button which when clicked, shows the variables and their values. See Fig. 12.9.
  - *Save Action* : Shows status of save action.
  - *Setup Action* : Shows status of setup action. This column is visible only in visual setup.
  - *Engine Run Action* : Shows status of engine run action.
  - *Post Processing Action* : Shows status of post process action. This column is visible only if any analyzer in the **Analyze** Window is marked to run automatically after each engine run.
  - *Engine Run Log* : Contains a button, which opens up a tab to view the log of the engine run.
- **Log Tab**: Maintains a log of steps and activities performed in the Parameter Scan Panel.



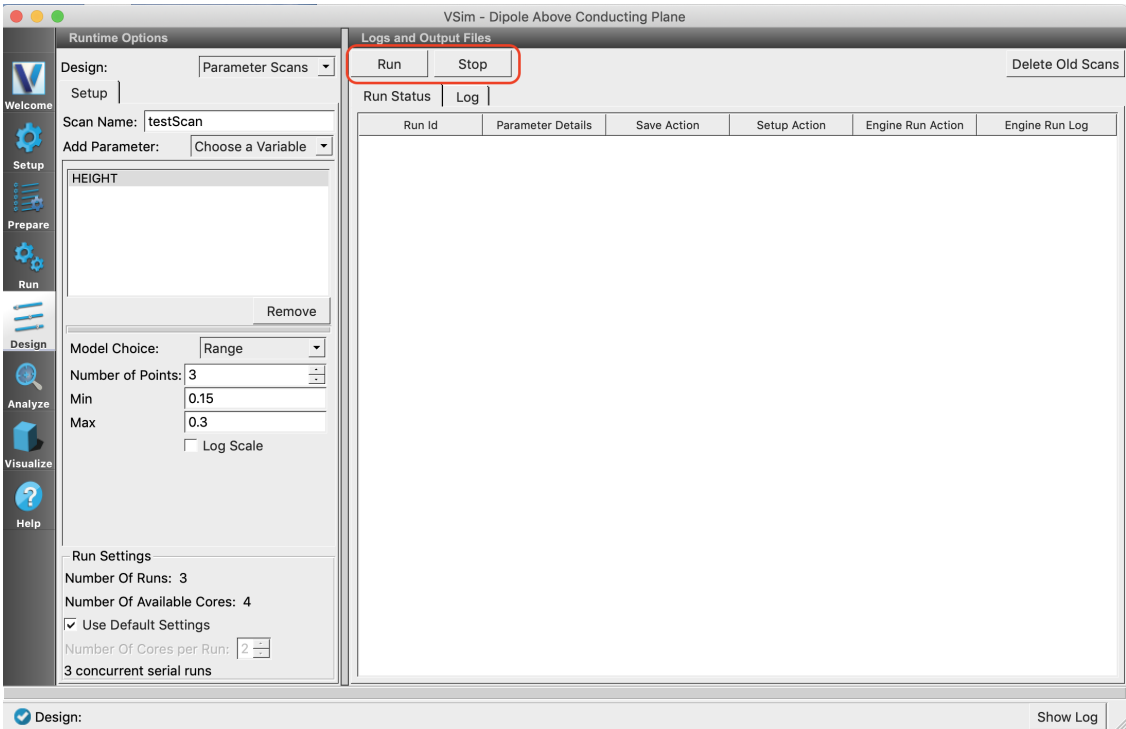


Fig. 12.7: Start and Stop

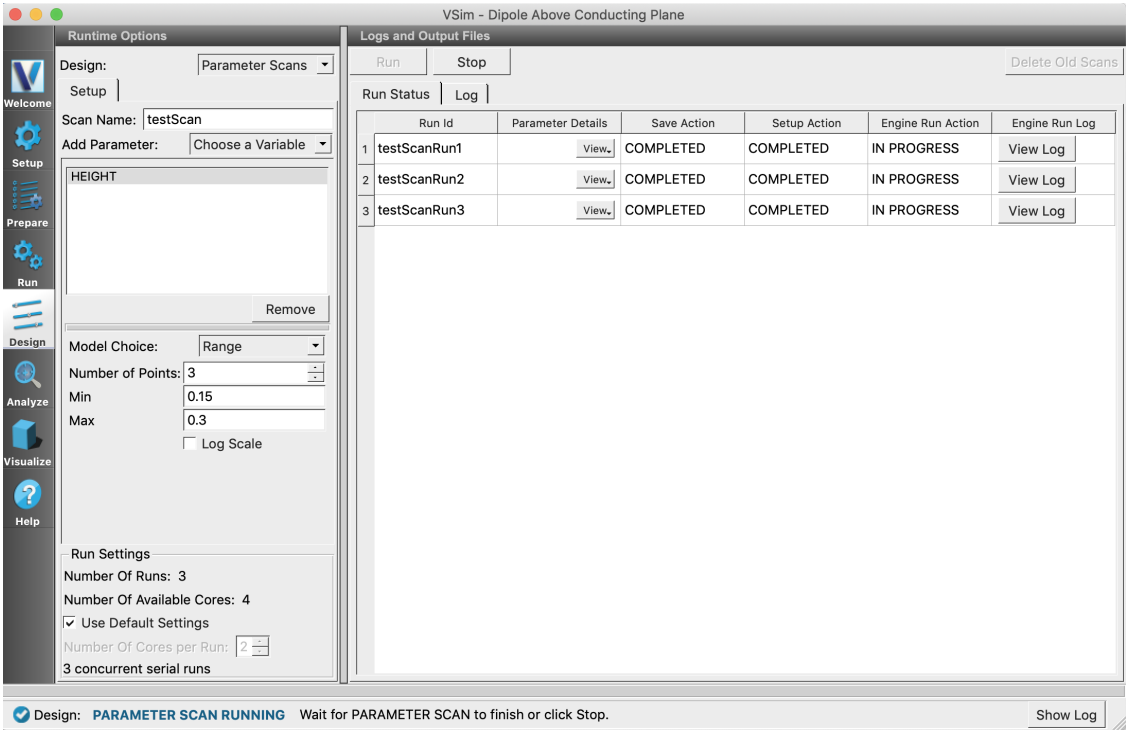


Fig. 12.8: Run Status Table

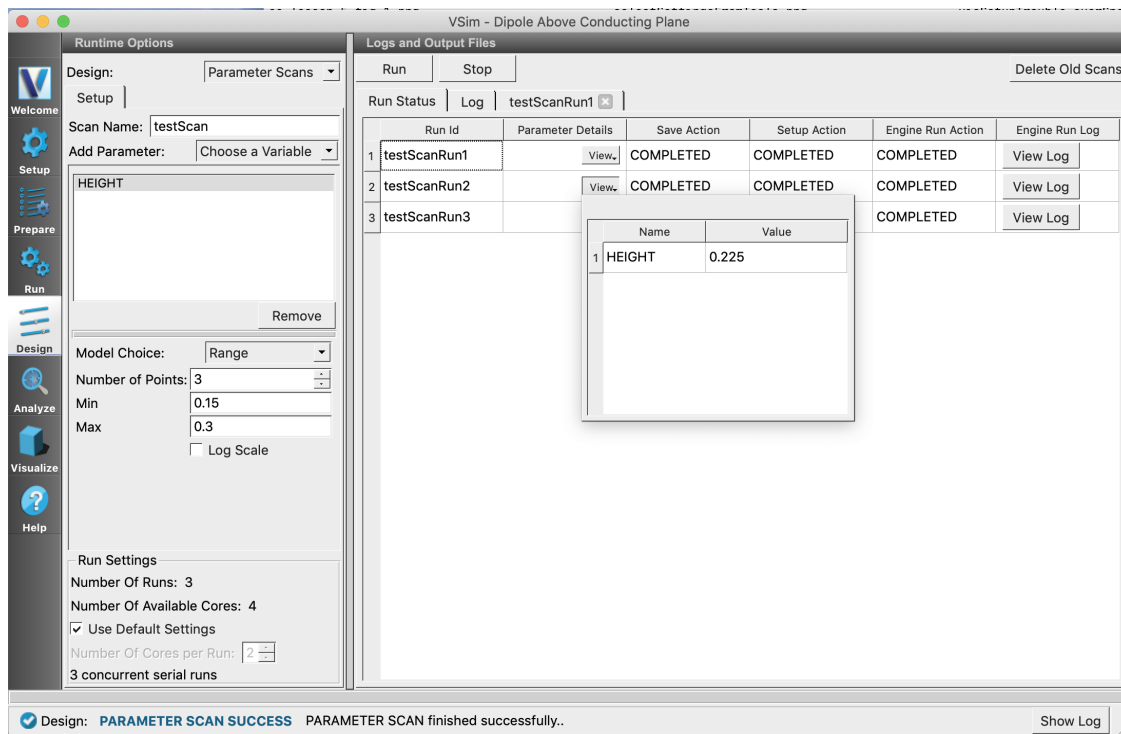


Fig. 12.9: View Variable and Value

## 12.7 Running an Analyzer on the Sub Simulation Data

Click on the **Analyze** icon after the completion of parameter scan. From the *Apply To* drop down, choose a sub simulation on which the analyzer is to be run. *Primary Run* refers to the data of the parent simulation. See Fig. 12.10.

If you want an analyzer to run automatically after each sub simulation engine run, make sure to check *Run Automatically After Engine* before starting the parameter scan.

## 12.8 Visualization of Sub Simulation Data

Click on the **Visualize** icon on the far left of VSimComposer Window. From the *Choose a Run* drop down, you will be able to choose the sub simulation whose data you want to visualize. *Primary Run* refers to the data of the parent simulation. Try Choosing *testScanRun1*. See Fig. 12.11.

On the next panel choose *Data Overview* from the *Add a Data View* drop down. The *View Parameters* button can be clicked to view the variables and their values that was used in the chosen sub simulation. See Fig. 12.12. Switching to the visualization of another sub simulation or the parent simulation can be done by choosing desired option from the *Choose a Run* drop down.

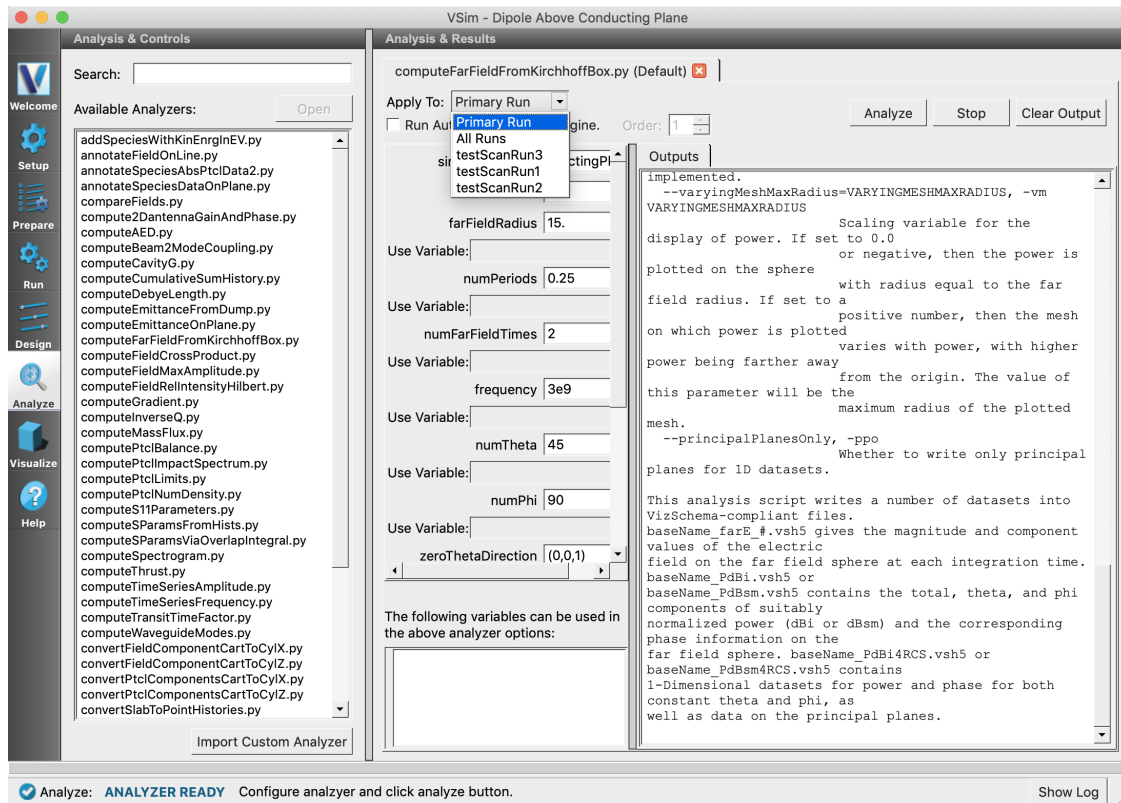


Fig. 12.10: Analysis of sub simulation data

## 12.9 Directory Structure

First a folder named `simulationName_scanName` is created in the parent simulation directory. This folder contains individual folders for each sub simulation and a test file named `simulationName_scanNameRunInfo.txt`. This file contains all the information (variables used, values, status) about the sub simulations. Folders for each sub simulation contains files and data related to the simulation. The directory structure for our **Dipole Above Conducting Plane** is shown in Fig. 12.13 .

## 12.10 Deleting Existing Parameter Scan Data

Clicking *Delete Old Scans* button, allows deletion of parameter scan files and folders from older runs.

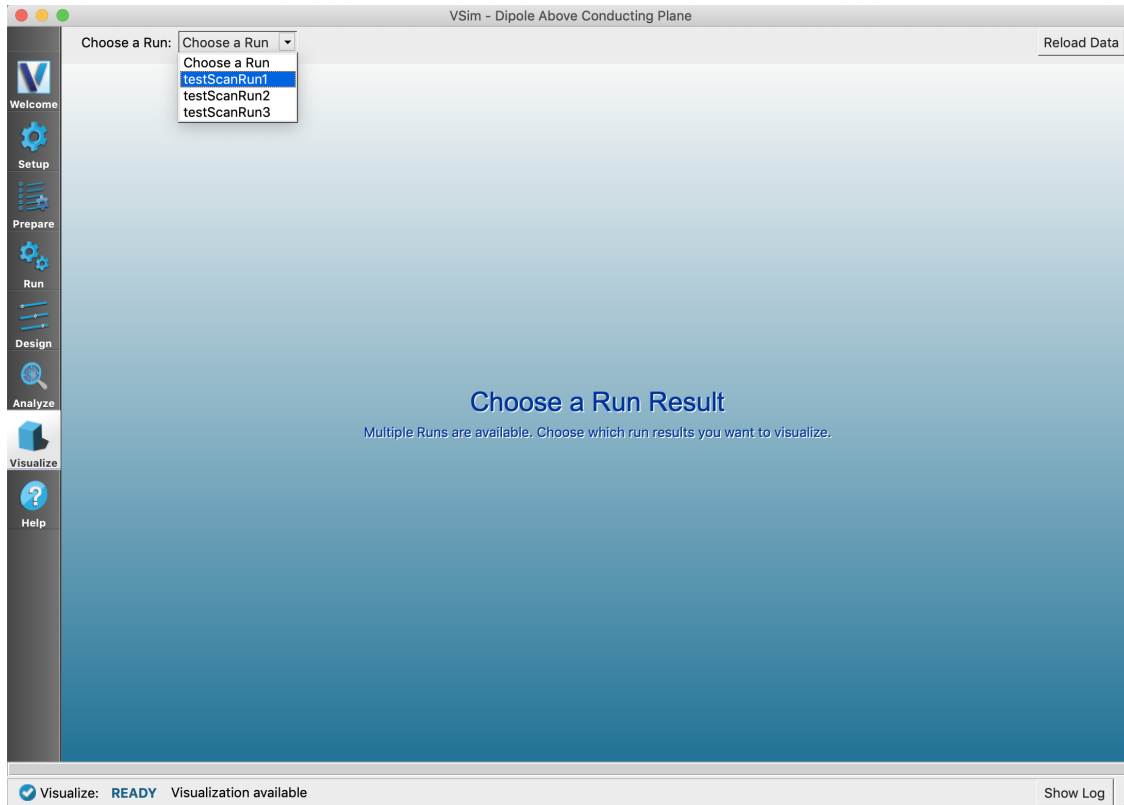


Fig. 12.11: Choose a sub simulation from the drop down

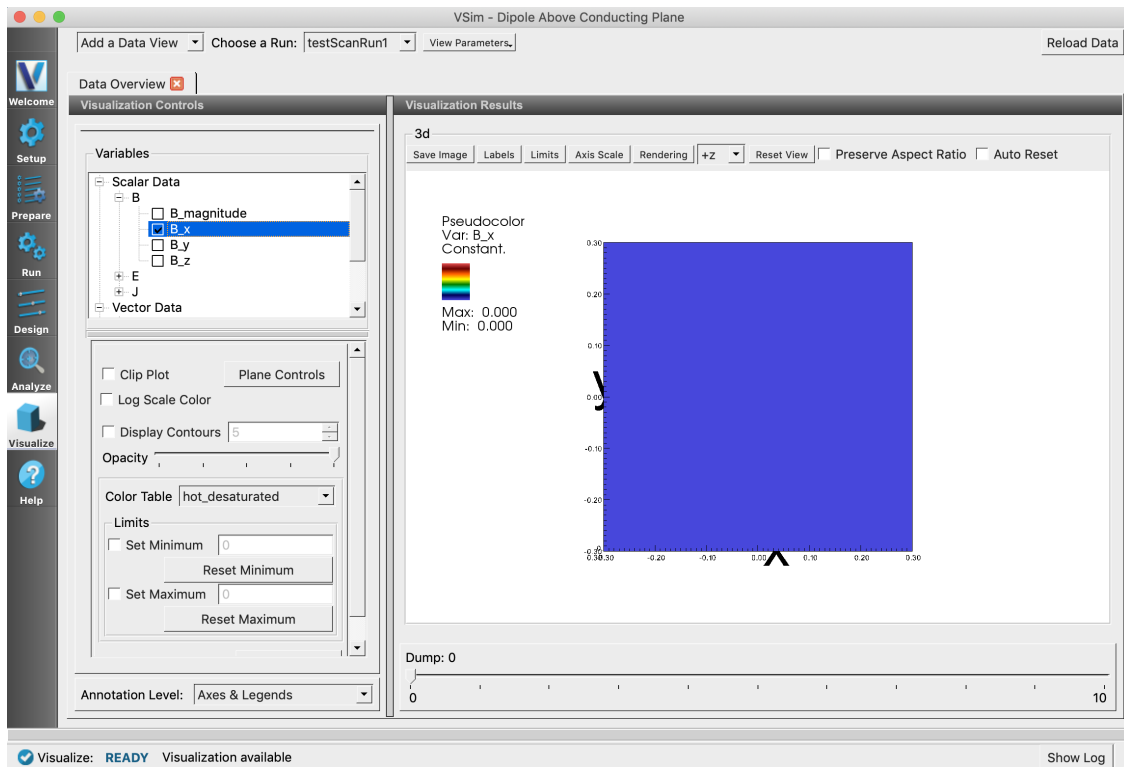


Fig. 12.12: Visualization of a sub simulation

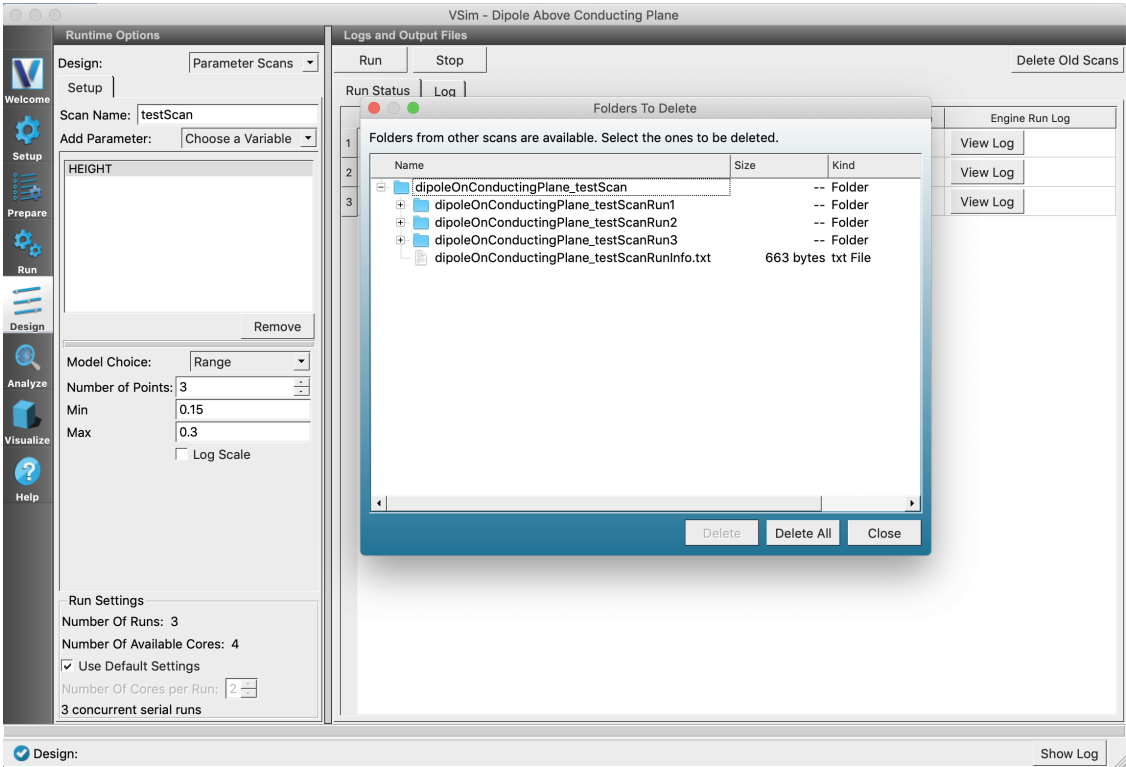


Fig. 12.13: Delete Old Dialog showing directory structure



## DATA ANALYSIS TAB

### 13.1 Overview of Using Analyzers

It is possible to run postprocessing analyzers within the GSimComposer environment. These analyzers can process data generated in a simulation and write that data to a .h5 file that can then be visualized like any other simulation data.

Here, we will discuss the basic process of using one of the many analyzers included with GSim. It is also possible to import custom analysis scripts. For more details on these, please see Creating Custom Python Analyzer Scripts section in the “Customization” Manual.

In this section, we will go through the basics of running analyzers, and will incorporate the two stream example (visual setup) to illustrate the processes.

#### 13.1.1 Using An Example To Demonstrate How to Use an Analyzer

First we open an example and run a simulation in order to get some data to analyze. Open the two stream example and follow along:

- Select the *New -> From Example* menu item in the *File* menu.
- In the resulting *Examples* window, expand the *GSim for Basic Physics* option.
- Expand the *Basic Examples* option.
- Select *Two-Stream Instability* and press the *Choose* button.
- In the resulting dialog, create a new folder if desired, and press the *Save* button to create a copy of this example in your run area.
- This will open the Two-Stream Instability example. For this exercise, we will use the default parameters.

See *Setup Window for Two Stream Example*.

#### 13.1.2 Select an Analyzer (Installed with GSim)

GSimComposer allows the user to select a analyzer from the **Analyze** window. You can type in the name of an analyzer to filter and shorten the list or double-click any analyzer in the list. You can also select the *Import Analyzer* button at the bottom to import your own customized script.

For the two stream example, wait until the simulation run has completed and then click on the **Analyze** tab located below the **Run** tab on the left hand side of the screen.

- In the *Analysis Controls* pane, a list of all of the analyzers will be available.
- For this example, we will select *computePtclNumDensity.py*.

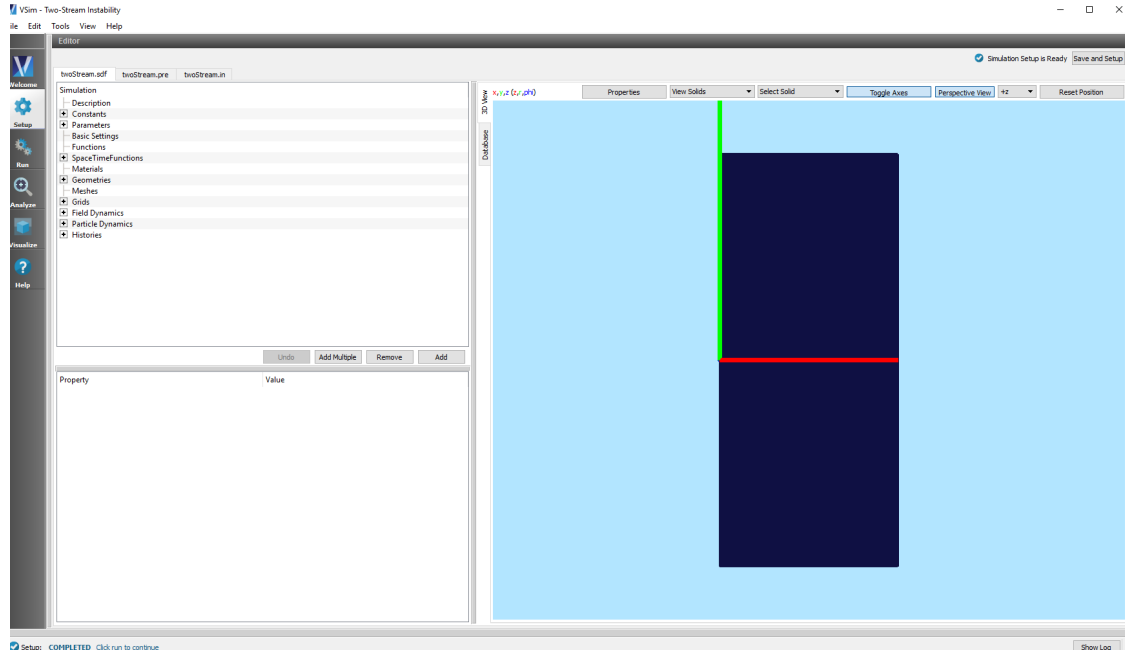


Fig. 13.1: Setup Window for Two Stream Example

- In the *Analysis Results* pane and under the *Outputs* tab, instructions as to which variables have to be given by the user will appear.
- For this particular analyzer, the only two variables are the *simulationName* and the *speciesName*. For *simulationName*, give it the name of the simulation, in our case *twoStream*. The *speciesName* parameter is either *electrons0* or *electrons1*, the two species of electrons in this simulation.

See [Analysis Window with Parameters](#).

This process should be the same for both visual and text-based simulations.

However, there are different processes for keeping analyzers available for your simulation through closing and reopening. For visual setup, simply run your simulation and analyzer of choice as usual, and then return to the **Setup** tab and select the *Save and Setup* button in the top right. This process saves your analyzer in the .sdf file for your simulation.

### 13.1.3 Running the Analyzer

Click the *Analyze* button located at the top right of the *Control* pane.

### 13.1.4 Output of an Analyzer

The data generated from the execution of an analyzer will be stored as a .vsh5 file and is visualizable underneath the **Visualize** tab. Be aware that the analyzer may also produce data that it writes to the screen or as hdf5 output.

Finishing up with the two stream example, the data we got from *computePtclNumDensity.py* can be visualized through the following:

- Open the **Visualize** tab. You may need to click on the *Reload Data* button at the top right of the *Visualization Controls* pane if you have gone back and forth between the windows and the GSim computational engine has generated more data.
- Switch the *Data View* drop-down menu at the top left of the pane to *1-D Fields*.



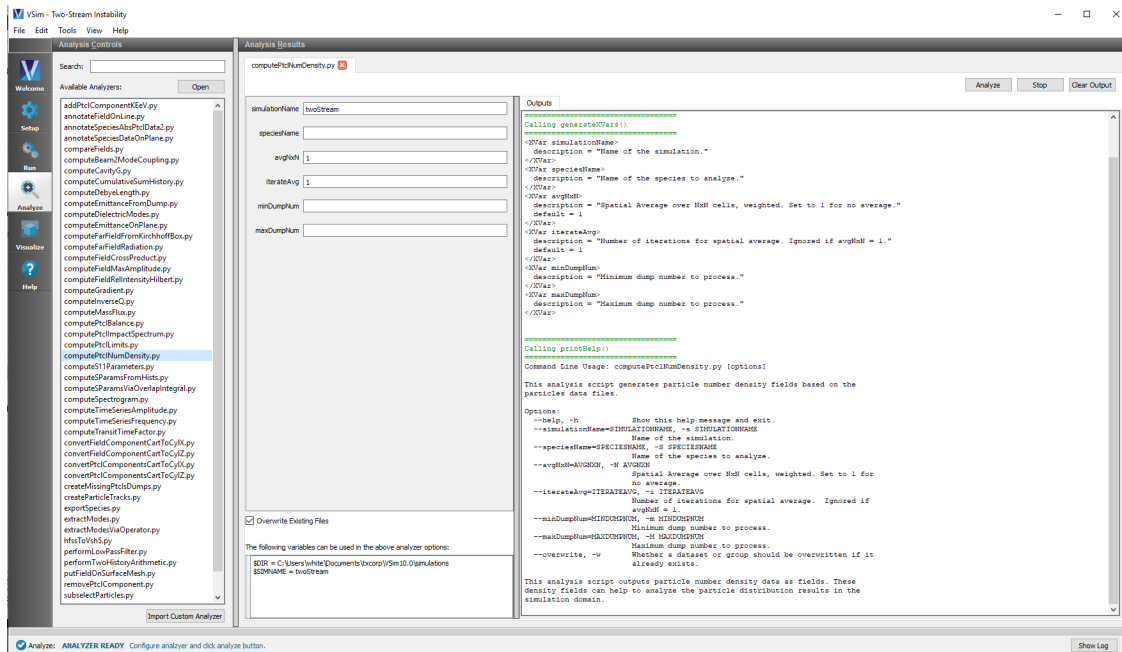


Fig. 13.2: Analysis Window with Parameters

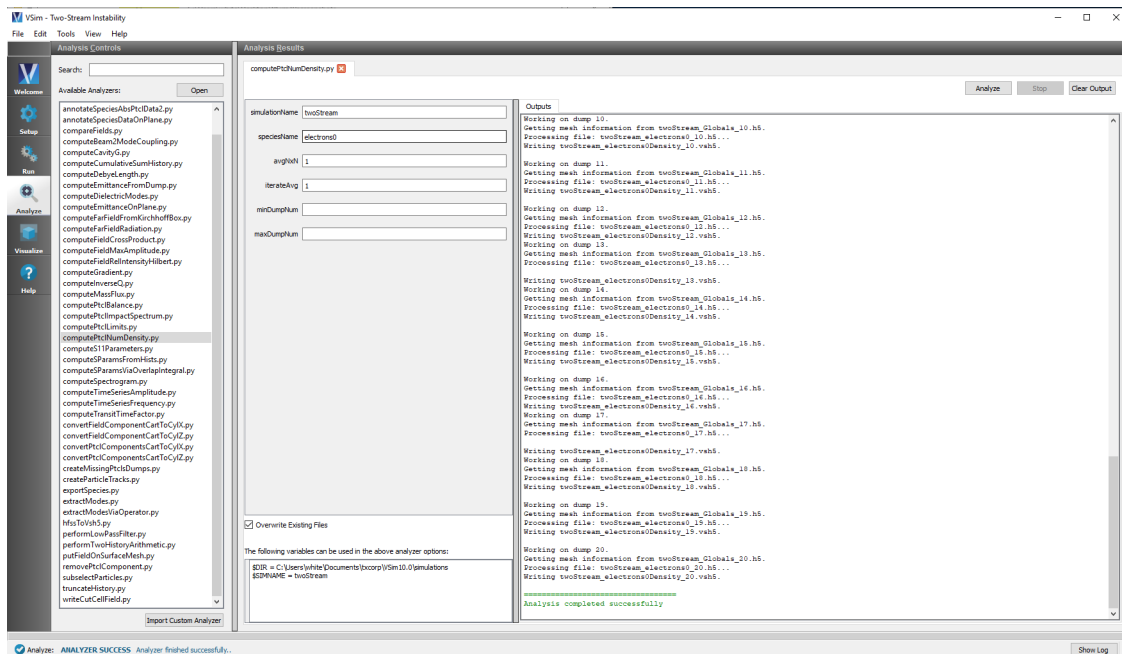


Fig. 13.3: Analysis Window Successful Run

- For *Graph 1*, select *electrons0Density* for the Base Variable from the drop-down menu under *Graph 1*.
- For *Graph 2*, select *electrons0Density* again for the Base Variable. In the *Visualization Results* pane (2nd graph), click the *Fourier Amplitudes* box to see the Fast Fourier Transform output of the frequency domain.
- To match the visualization of this documentation, select *Dump 2* in the slide bar at the bottom of the *Visualization Results* pane.

You can explore other visualization options in this window, or can rerun the simulation with different parameters to investigate further.

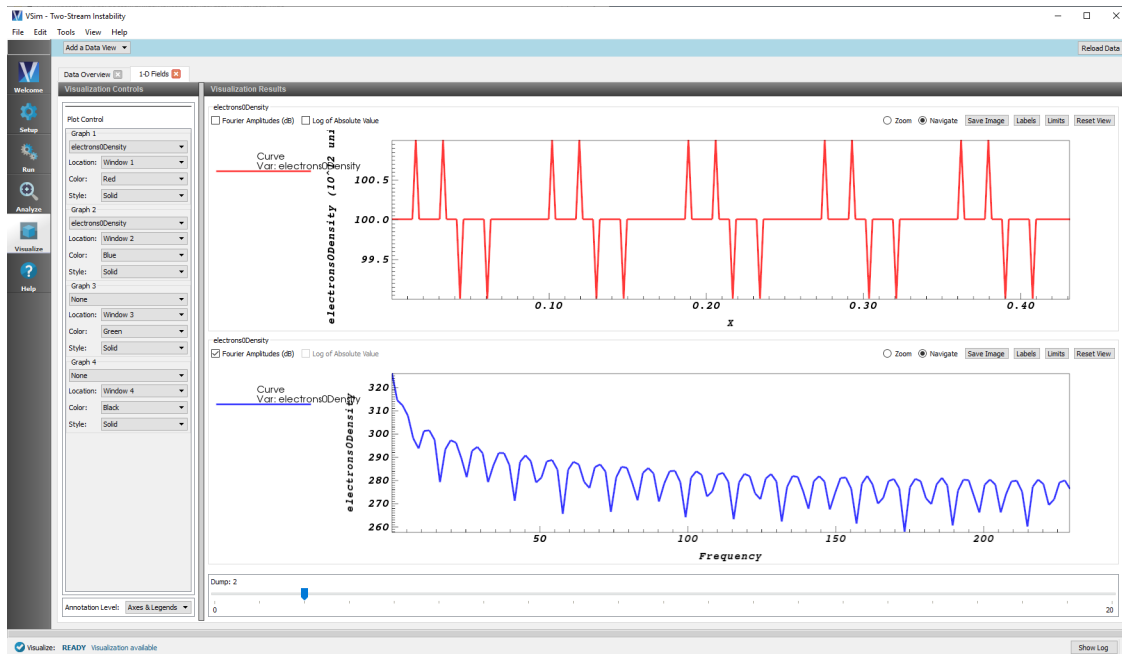


Fig. 13.4: Visualization of the analysis data

## VISUALIZATION TAB

### 14.1 Introduction to the Visualize Window

GSimComposer's Visualization feature is a flexible and comprehensive model viewer based on VisIt. The simulation tutorials and examples in *GSim Examples* provide several examples of using the Visualization feature's options in context.

The GSimComposer visualization tool is context sensitive, meaning that only those features that can be used with the current data are made available from the interface.

For more information on VisIt, please see: <https://wci.llnl.gov/codes/visit/> and [http://www.visitusers.org/index.php?title=VisIt\\_Wiki](http://www.visitusers.org/index.php?title=VisIt_Wiki)

For more information on using the VisIt context menu see: *Tools/Composer Menu: Visualization Options*.

The Visualization window is divided into a *Visualization Controls* pane on the left and a *Visualization Results* pane on the right.

The type of data available to view is governed by the *Data View* pull down menu.

### 14.2 Select the Visualize Icon from the Icon Panel

Upon successful completion of the simulation run, the last message in the *Engine Log* tab is a reminder that you can now select the **Visualize** icon from the icon panel on the far left of the GSimComposer window as seen in *Visualize Icon in Icon Panel*.

### 14.3 Data View Pull-down Menu

In the top left of the main pane, you may select the kind of visualization that is to be performed. Again, this menu is context sensitive, so not all options may be available for your simulation. For example, you may only choose phase space if you have particles, and you may only paint fields onto surfaces if you have complex boundaries specified in *GridBoundary* blocks.

- Data Overview
- 1-D Fields
- Field Analysis
- History
- Phase Space

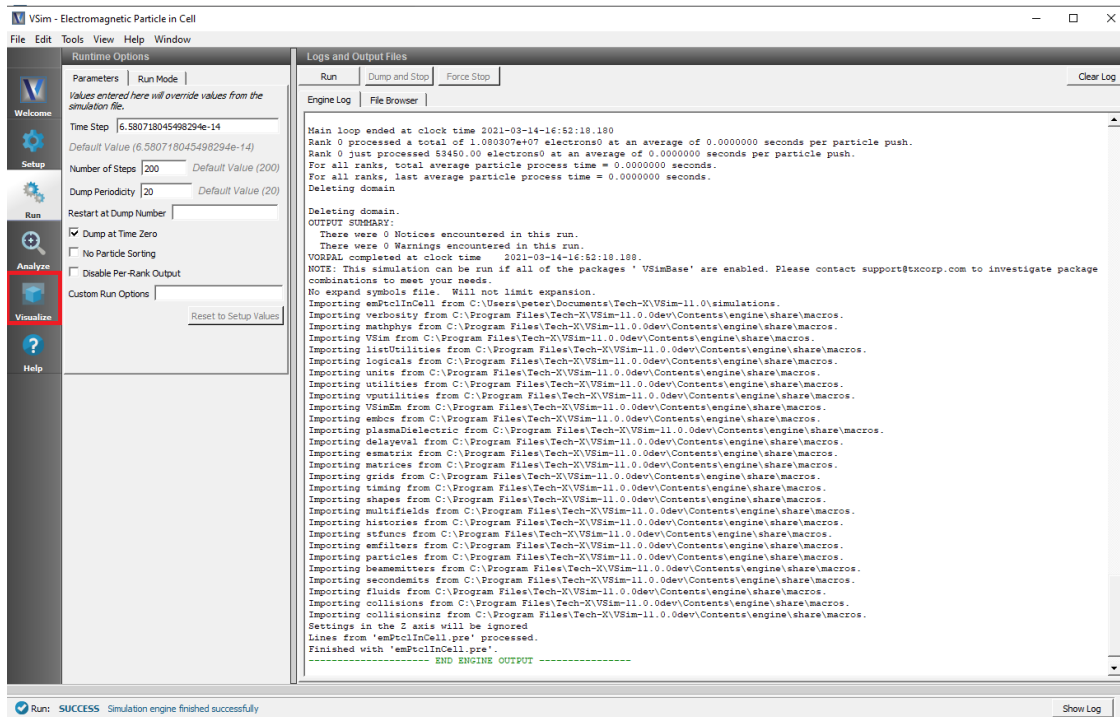


Fig. 14.1: Visualize Icon in Icon Panel

- Binning
- Paint Fields

See *Data View Menu*.

## 14.4 Standard Controls Available Across Multiple Views

Several control buttons or choices are available across several different *Data Views*. They have the same functionality in each case and are documented below.

### Annotation Level

To adjust the Annotation Level, use the *Annotation Level* drop-down menu at the lower left of the *Visualization Controls* pane.

- No annotations
- Axes only
- Axes & Legends
- All annotations

See *Annotation Level*.

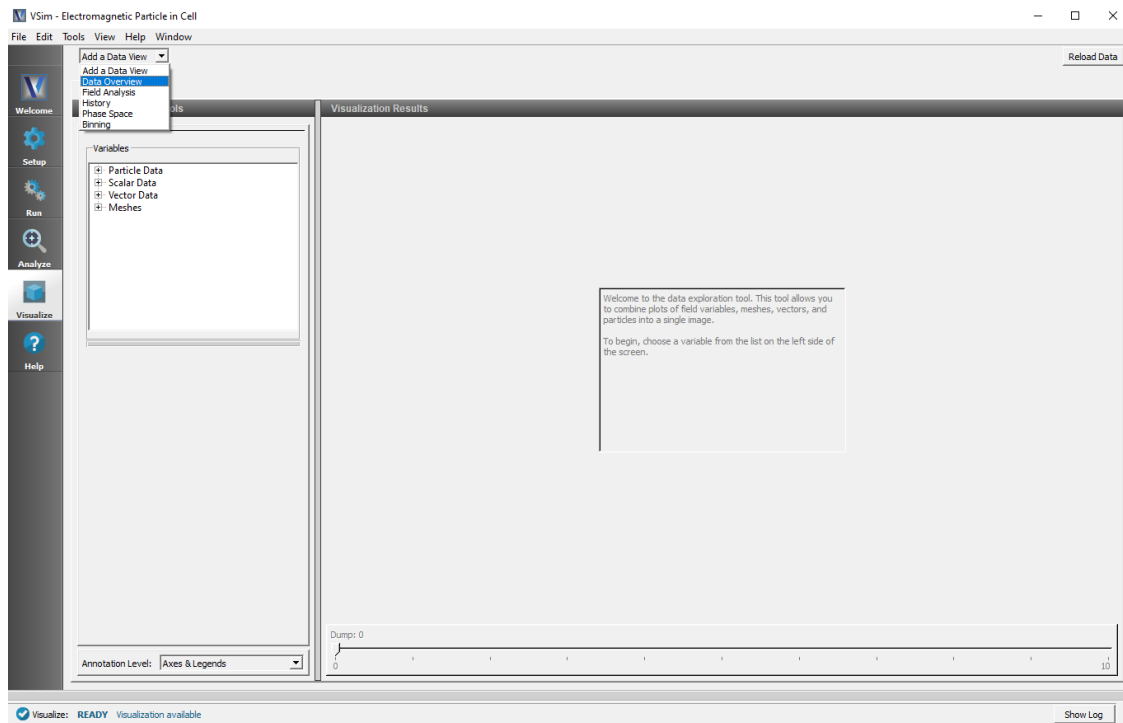


Fig. 14.2: Data View Menu

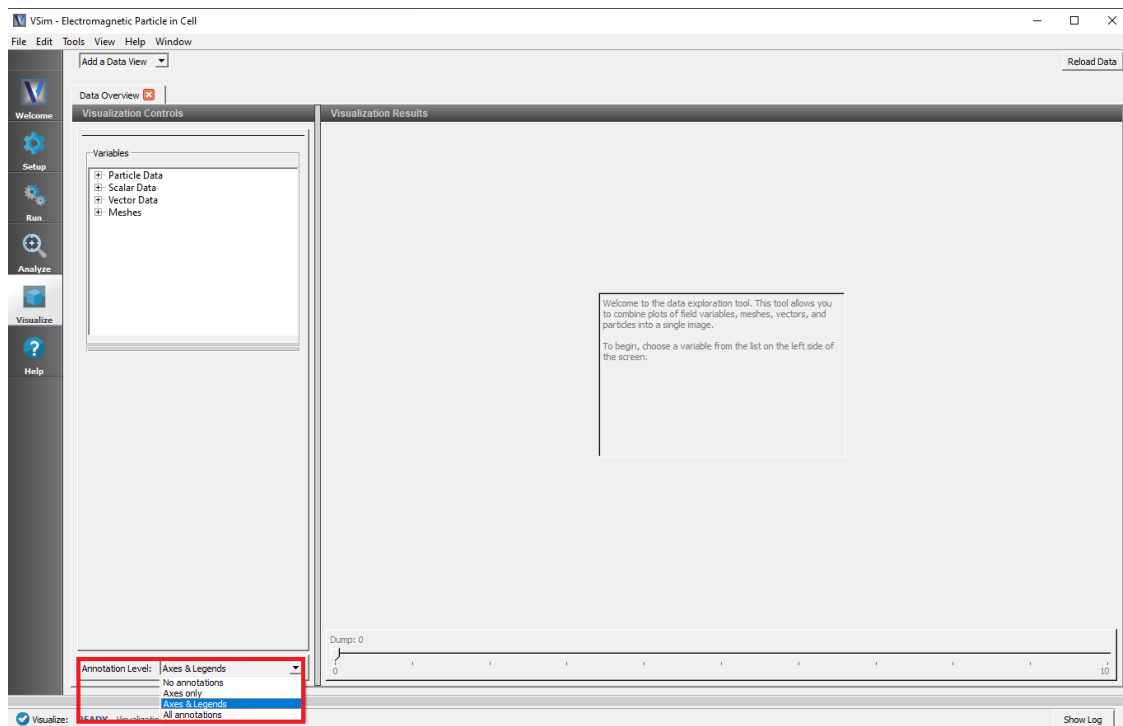


Fig. 14.3: Annotation Level

## Reload Data

You can visualize data from a simulation run as soon as it becomes available. If you decide to visualize data before a run is complete by switching to the **Visualize** tab, the GSim engine continues creating data files in the background. Later, when more data is available for visualization or the simulation run is complete, use the *Reload Data* button in the top right-hand corner to visualize the new data. See *Controls Pane Buttons*.

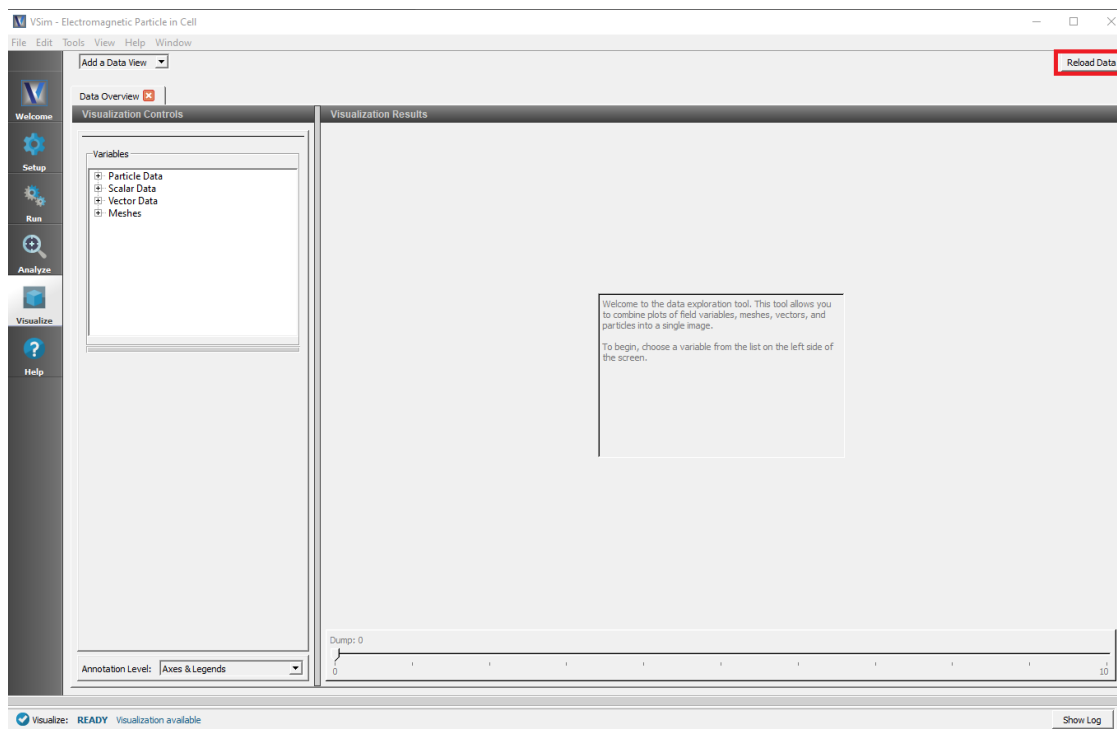


Fig. 14.4: Controls Pane Buttons

## Save

- *Save Image* This option saves the current image to your computer. You will be given options on where to save, the file name, and format as well as some options on size and dimension.
- *Save Movie* This option brings up the *Save Movie* dialog and an mpg movie can be created from the visualized variables. See *Save Movie Dialog*.

The first page allows setting of the movie size. Default window can be scaled or custom height and width can also be set. The second page allows setting of movie attributes like frames per second, first frame, last frame and stride. The third page can be used to set the name and location of the movie. Click on *Done* to create the movie.

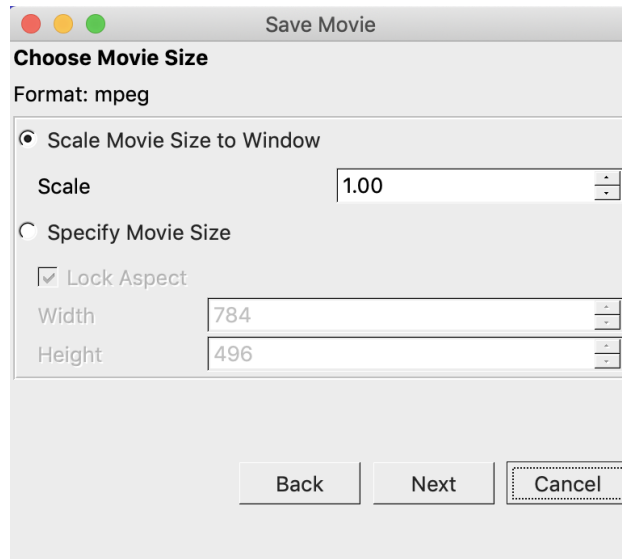


Fig. 14.5: Save Movie Dialog

## Labels

This brings up the *Axis Labels* window. See [Axis Labels Menu](#).

## Limits

This button enables adjusting the *Limits* for each axis. See [Axis Limits Menu](#).

## Rendering

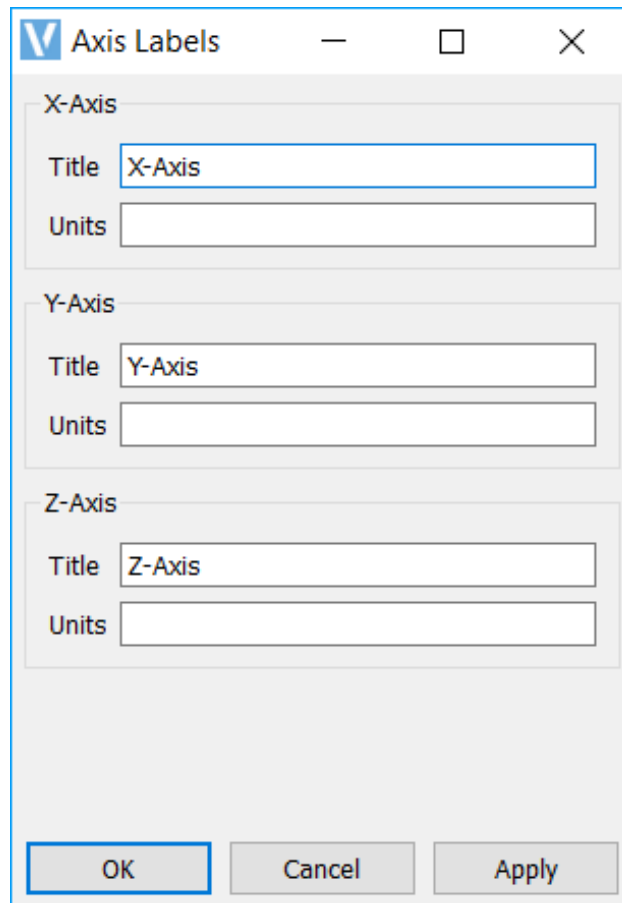
This button allows for adjustment of the lighting and stereo effects. See [Rendering Menu](#).

## Reset View

This button returns the objects in the *Visualization Results* pane back to their original location.

## Auto Reset

Checking the *Auto Reset* box will force a *Reset View* each time the dump slider is moved.



The image shows a dialog box titled "Axis Labels" with a blue icon on the left and standard window controls (minimize, maximize, close) on the right. The dialog is organized into three sections: "X-Axis", "Y-Axis", and "Z-Axis". Each section contains a "Title" text field and a "Units" text field. The "X-Axis" section has "X-Axis" in the title field and is empty in the units field. The "Y-Axis" section has "Y-Axis" in the title field and is empty in the units field. The "Z-Axis" section has "Z-Axis" in the title field and is empty in the units field. At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply". The "OK" button is highlighted with a blue border.

Axis	Title	Units
X-Axis	X-Axis	
Y-Axis	Y-Axis	
Z-Axis	Z-Axis	

OK Cancel Apply

Fig. 14.6: Axis Labels Menu



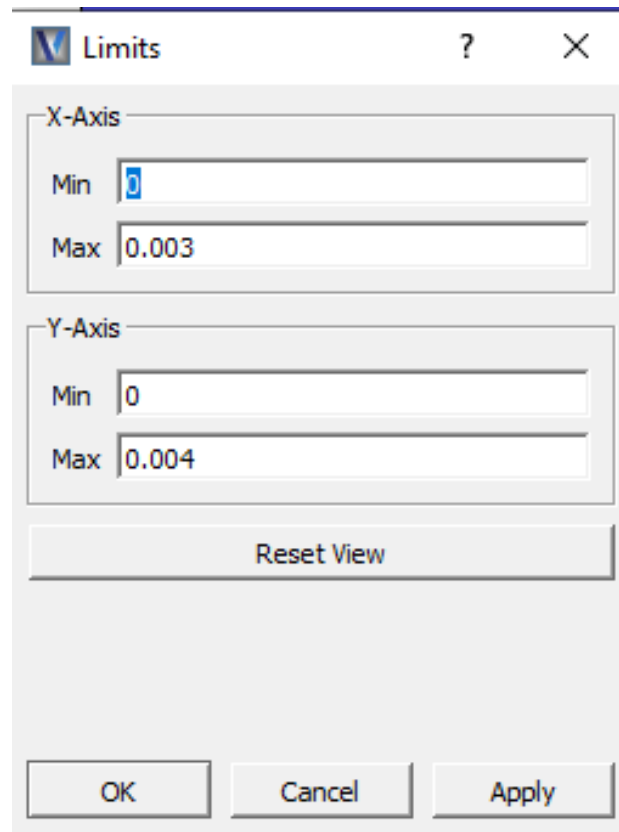


Fig. 14.7: Axis Limits Menu

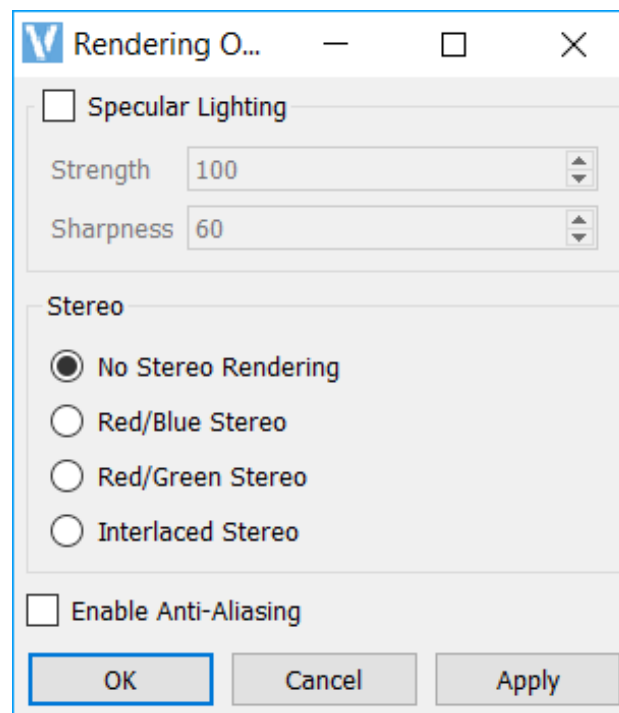


Fig. 14.8: Rendering Menu

## Dump Slider

The slider at the bottom of the *Visualization Results* pane allows the user to move through the simulation results in time. Only the times for which files were “dumped” can be viewed.

## 14.5 Data Overview

### Variables

The *Variables* section of the *Visualization Controls* pane enables you to choose which aspects of the simulation data to visualize. The types of variables that are available in the *Variables* section are dependent on your particular simulation. Below are some typically available types of variables for a 3D simulation containing fields and particles.

#### Particle Data

Types of *Particle Data* may include any *Species* in the input file:

- electrons
- ions
- neutrals

The base *species name* variable will show the particle position. The *\_x*, *\_y*, *\_z* (or *\_r \_z*) variable will show the particle at it's position, with the value assigned being the coordinate in the respective direction.

The *\_ux*, *\_uy*, *\_uz*, *\_ur*, *\_uphi*, variables will also plot the particle position, with the value assigned being the relativistic velocity in that axial direction.

If the simulation has been run in cylindrical coordinates, a {particleSpecies}\_transform set of variables will be available. At this time the *\_transform* variables will be equivalent, this transformation is used for non-uniform cylindrical grids, which are not currently available in visual setup.

#### Scalar Data

Types of *Scalar Data* may include fields like:

- E
- B
- J

#### Vector Data

Types of *Vector Data* include:

- E
- B
- J

#### Meshes

Types of *Meshes* include:

- globalGridGlobal (B)
- globalGridGlobal (E)
- globalGridGlobal (J)
- globalGridGlobal (coax)

See *Particle, Scalar, Mesh Data Variables*.

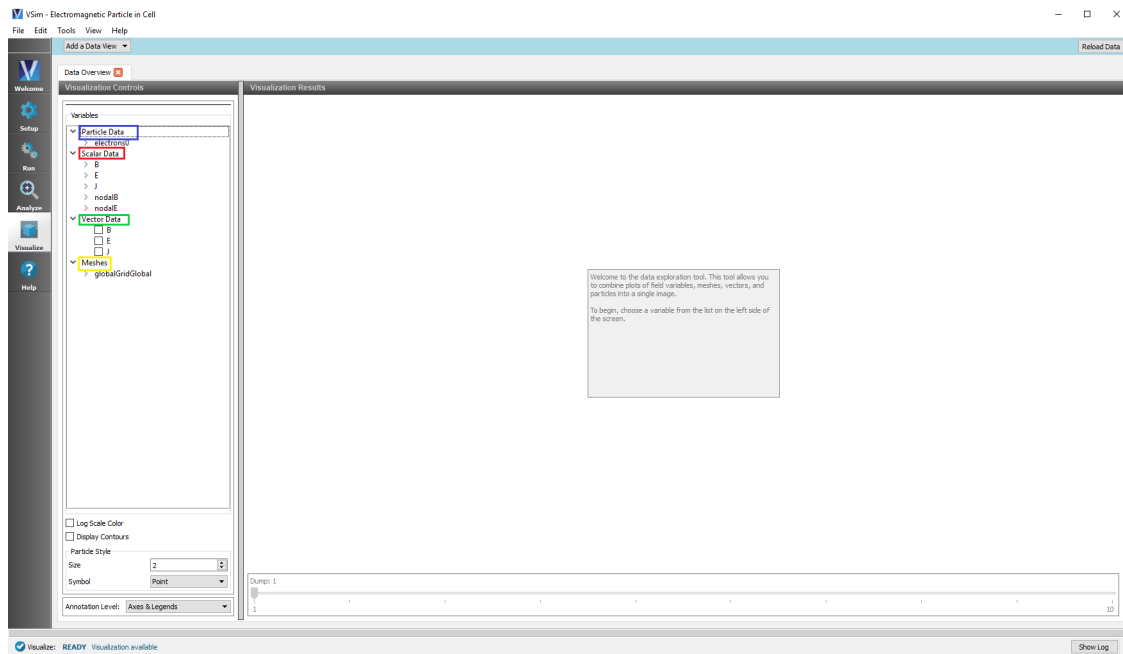


Fig. 14.9: Particle, Scalar, Mesh Data Variables

## Geometries

Types of *Geometries* include:

- poly A wireframe of the part
- poly\_surface A solid view of the part
- MappedPolysData A wireframe of the part
- MappedPolysData\_surface A solid view of the part
- PecShapes All geometries in the simulation assigned PEC Material

All geometries can be clipped simultaneously by selecting the *Clip plot* checkbox, with the plane set by selecting *Plane Controls*. It is also possible to change the opacity of a geometry using the *Opacity* slider.

See *Geometry Data Variables*.

### 14.5.1 Context Sensitive Options

The below options are available to each individual selection of scalar, particle or vector data. They may be applied to all pieces of scalar/particle data simultaneously by being set in the *Scalar Data* tab, or they can be applied on an individual basis, allowing for different settings for each piece of concurrently visualized data.

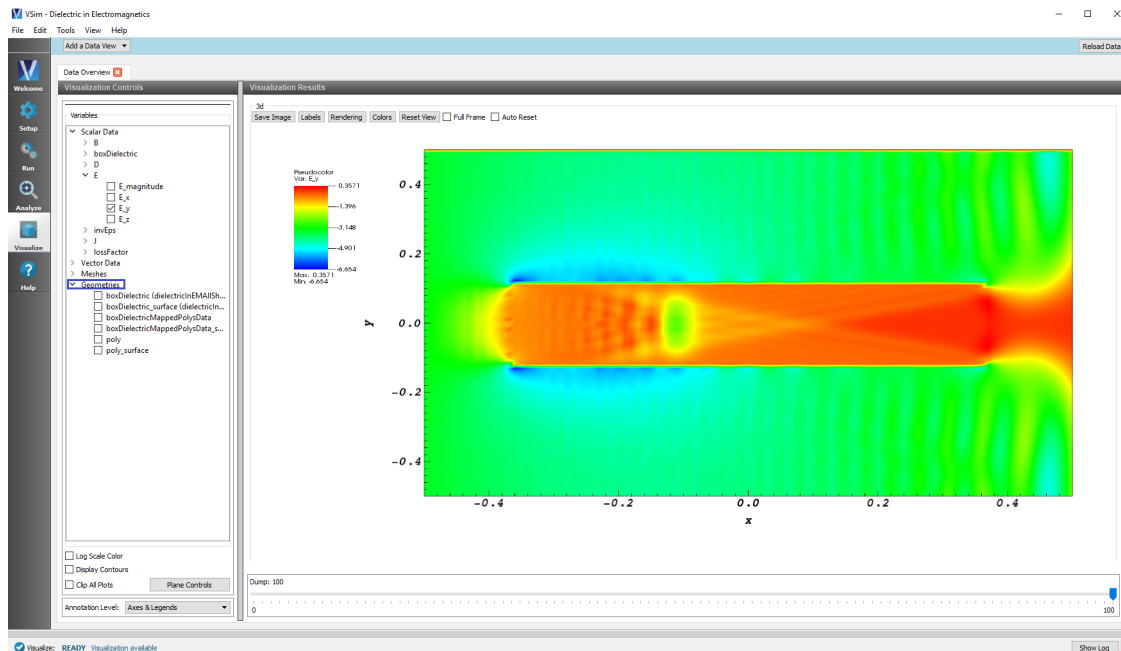


Fig. 14.10: Geometry Data Variables

## Log Scale Color

If the appropriate field is selected, the *Log Scale Color* checkbox will be available to enable and disable display of log scale color.

Checking this box will put the color on a log scale. This is useful to see details in a field. See [Log Scale Color Checkbox](#).

## Display Contours

The default value in the *Contours* field is 5. If you select the Display Contours check box and have an appropriate data set selected, the number of contours can be changed to any positive number. See [Contours](#).

## Clip Plot Checkbox

If the appropriate data is selected, the *Clip Plot* checkbox will be available to enable and disable plot clipping.

When *Clip Plot* is checked, the *Plane Controls* can be used to select an axis intercept or normal vector for the clipping. See [Clip Plot Checkbox](#).

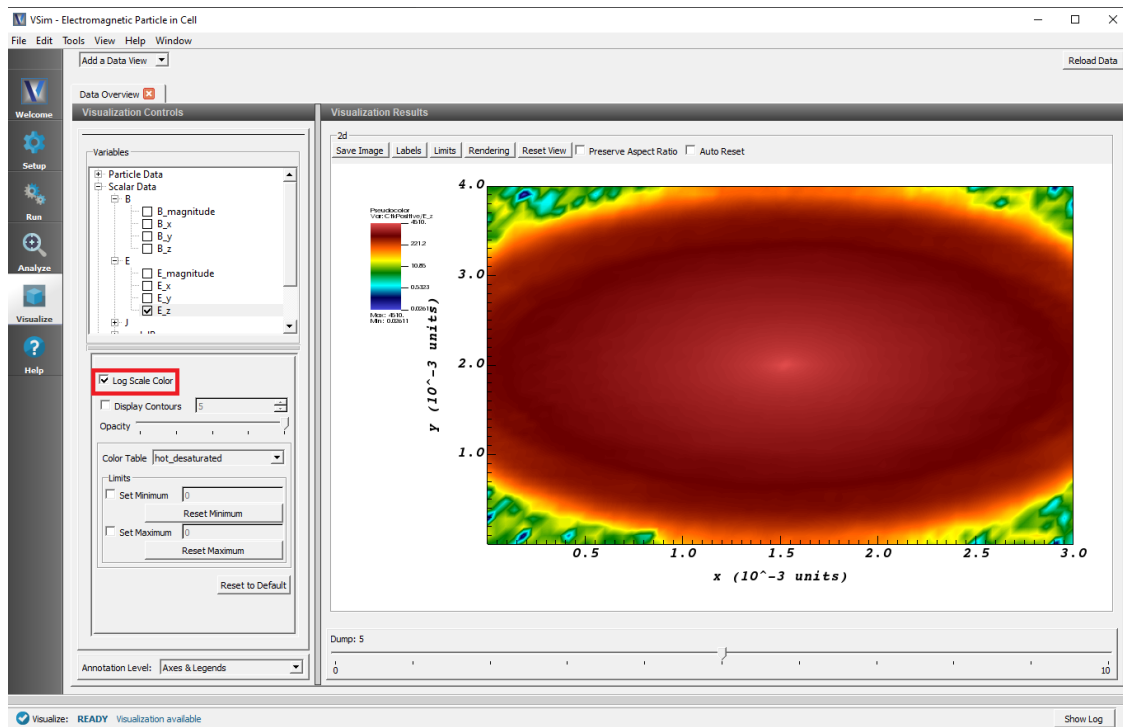


Fig. 14.11: Log Scale Color Checkbox

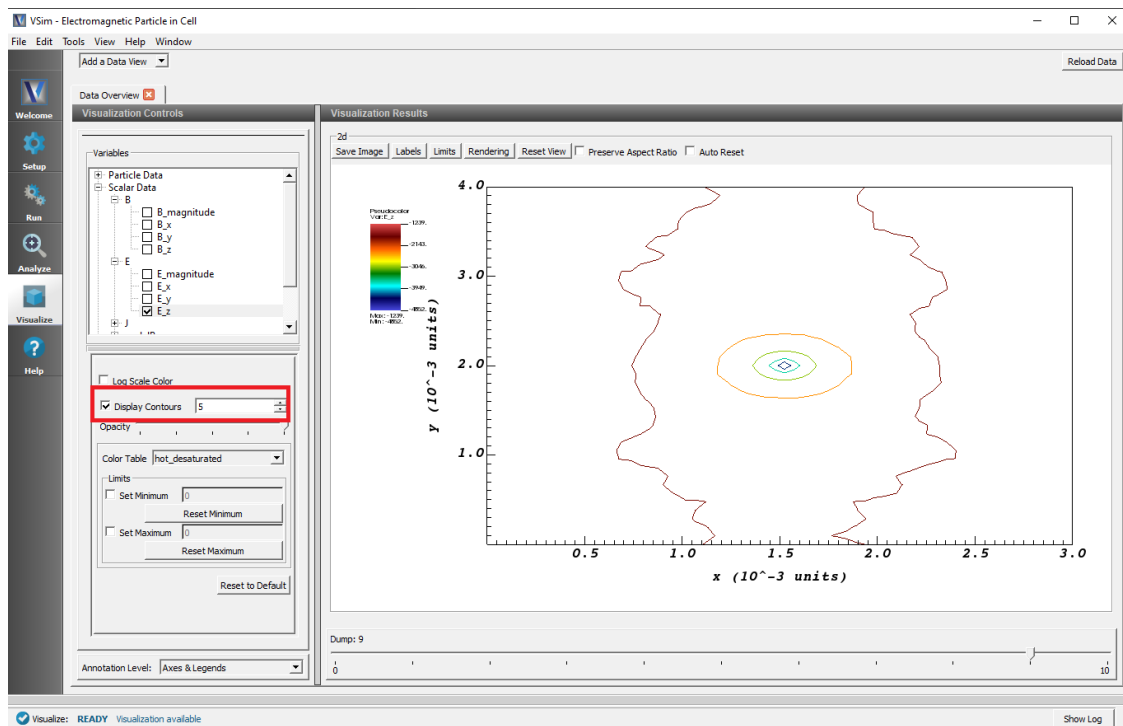


Fig. 14.12: Contours

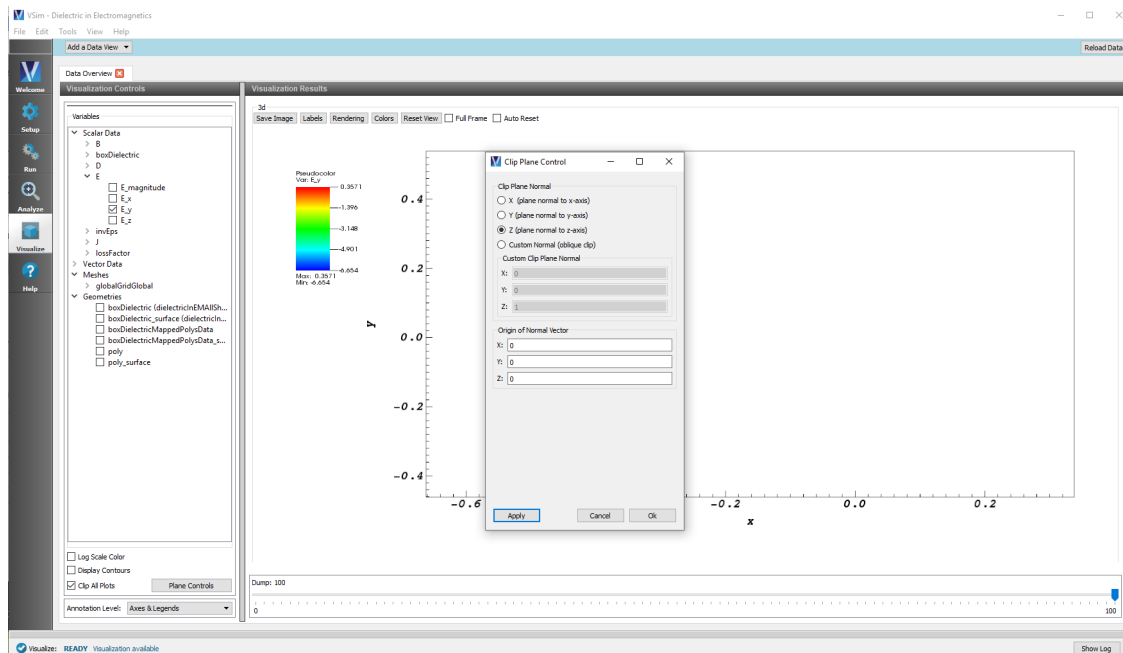


Fig. 14.13: Clip Plot Checkbox

## Opacity

The opacity of any object in the visualization window can be adjusted, sliding the *Opacity* slider to the left will reduce the opacity, to the right will increase the opacity.

## Color Table

The color table can be selected for each piece of visualized data, though *hot\_desaturated* is the standard.

Minimum and maximum values can be set by hand, by default they are scaled to the minimum and maximum values of that dataset.

## Particle Style

The *Particle Style* section of the *Visualization Controls* panel enables you to control the appearance of particles in the visualization.

The *Size* field contains the size of each particle symbol.

The *Symbol* pulldown menu contains choices for the following shapes:

- Point (default)
- Box
- Axis
- Icosahedron
- Octahedron
- Tetrahedron

- Sphere Geometry
- Sphere

See *Particle Style*.

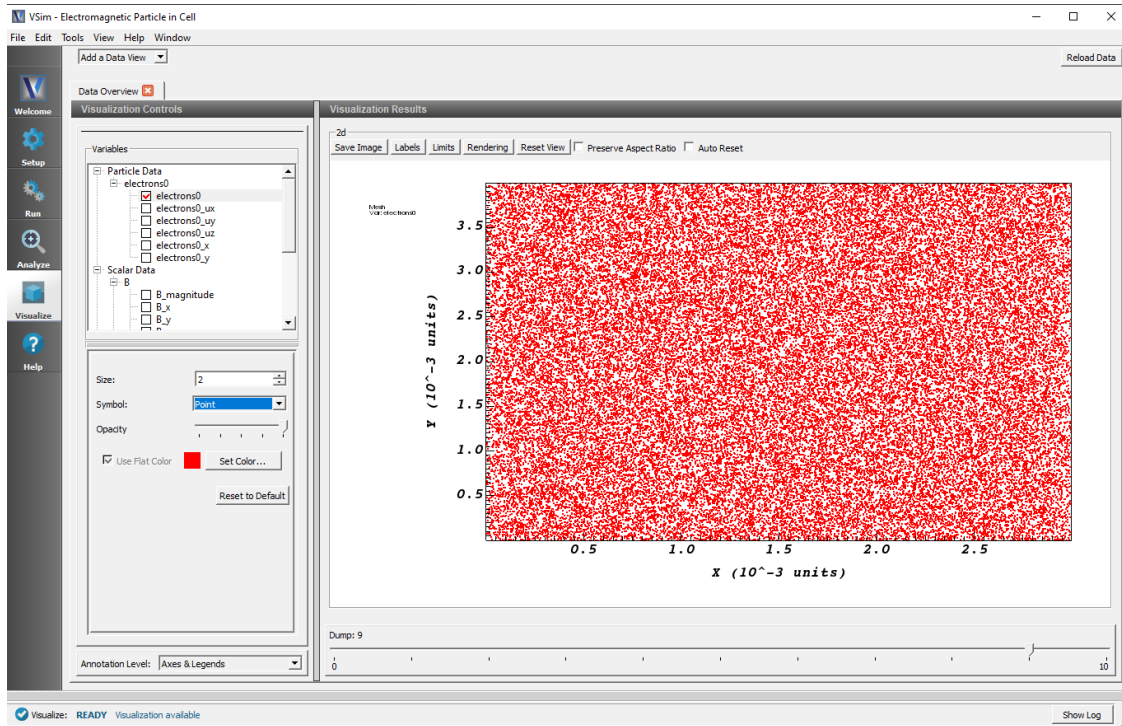


Fig. 14.14: Particle Style

It is also possible to set a specific flat color for the particles, selected with the *Set Color* button.

## 14.6 Field Analysis

The *Field Analysis* data view will allow further analyzing of a particular field. From the *Add Data View* drop-down menu in the upper left corner of the pane, select *Field Analysis*. The *Visualization Controls* pane allows for selection of the field and selecting the line out location. The *Visualization Results* pane contains a 2d view of the chosen field, and a lineout at the selected location.

### Field

The Field drop-down menu will allow you to choose which field from your simulation to do further analysis on. See *Field drop down*.

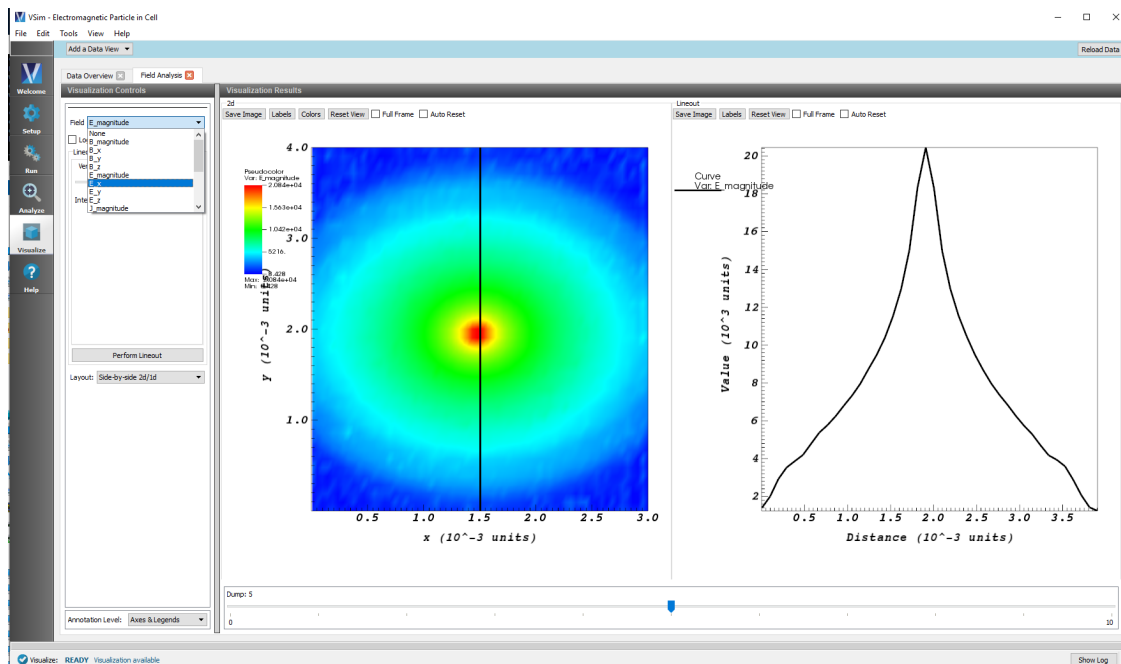


Fig. 14.15: Field drop down

## Slice Settings

Slice settings will appear when a 3D field is viewed. The slice settings allow you to set the position of a slice of the 3D field to create a 2D plot. The *Plane Controls* bottom allows for further control, including creating a slice at an angle. See [Slice settings and plane controls for a 3D plot](#).

## Log Scale Color Table

Checking this box will put the color on a log scale. This is useful to see details in a field. See [Log Scale Color Table Checkbox](#).

## Lineout Settings

The position of the lineout can be changed using the *Lineout Settings*. The lineout can easily be set to vertical or horizontal at a specified intercept location, or the *Advanced* tab can be used to set a line in any arbitrary direction or length.

After setting the desired location of the lineout, press the *Perform Lineout* button to replot the lineout. See [The lineout settings controls](#).



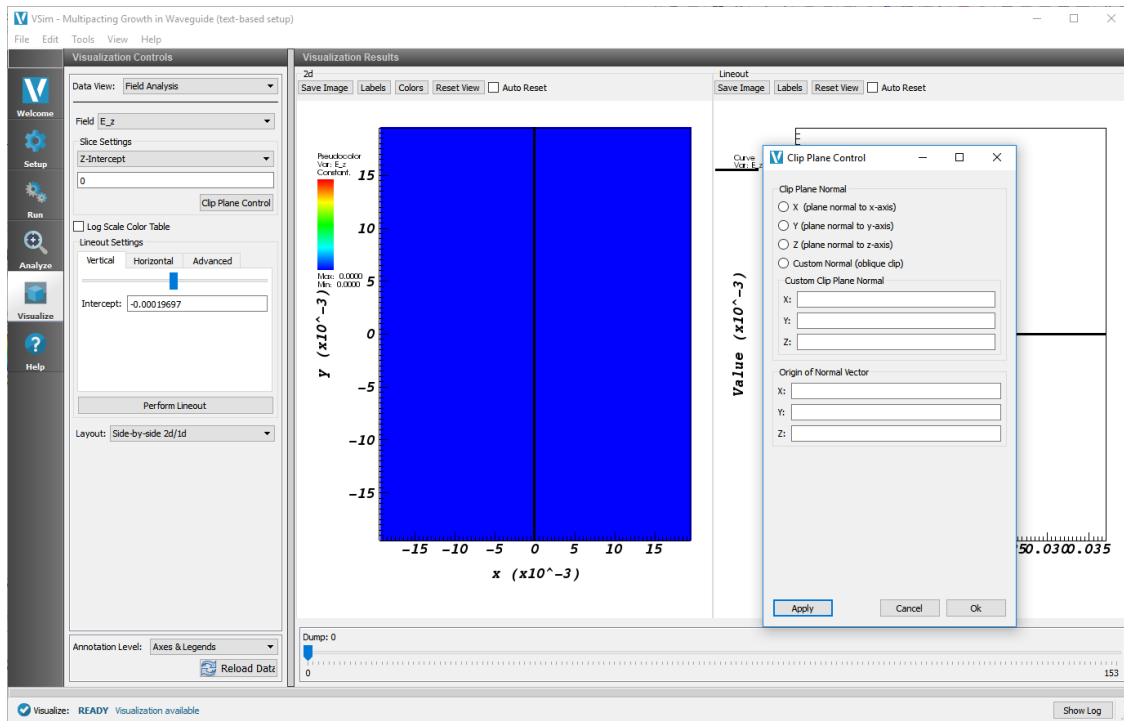


Fig. 14.16: Slice settings and plane controls for a 3D plot

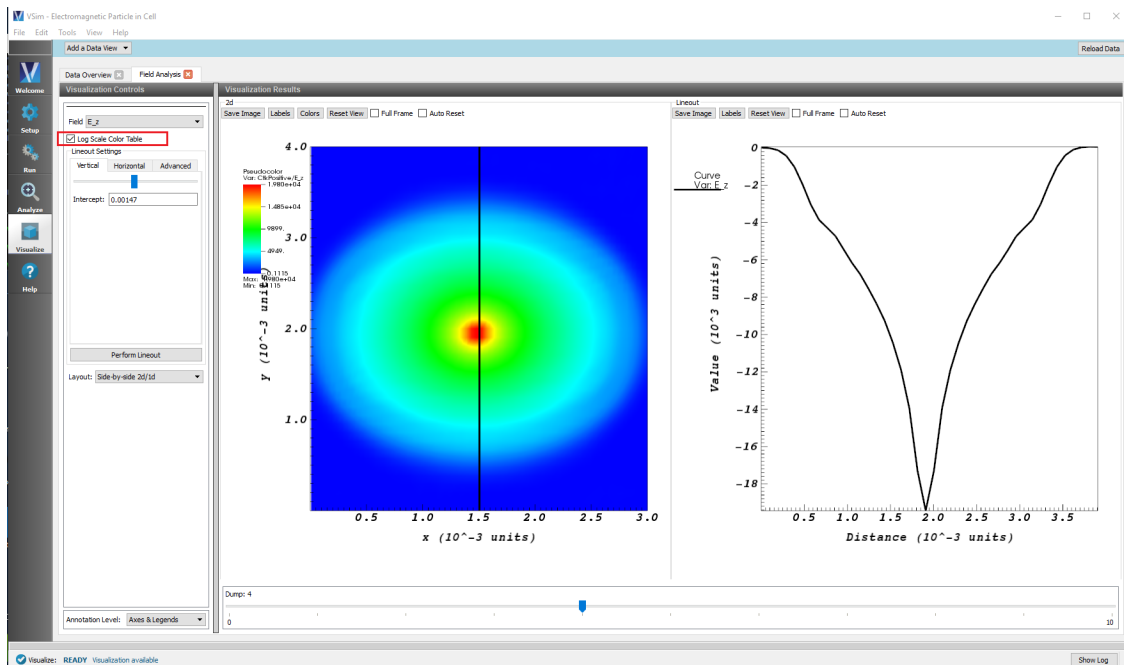


Fig. 14.17: Log Scale Color Table Checkbox

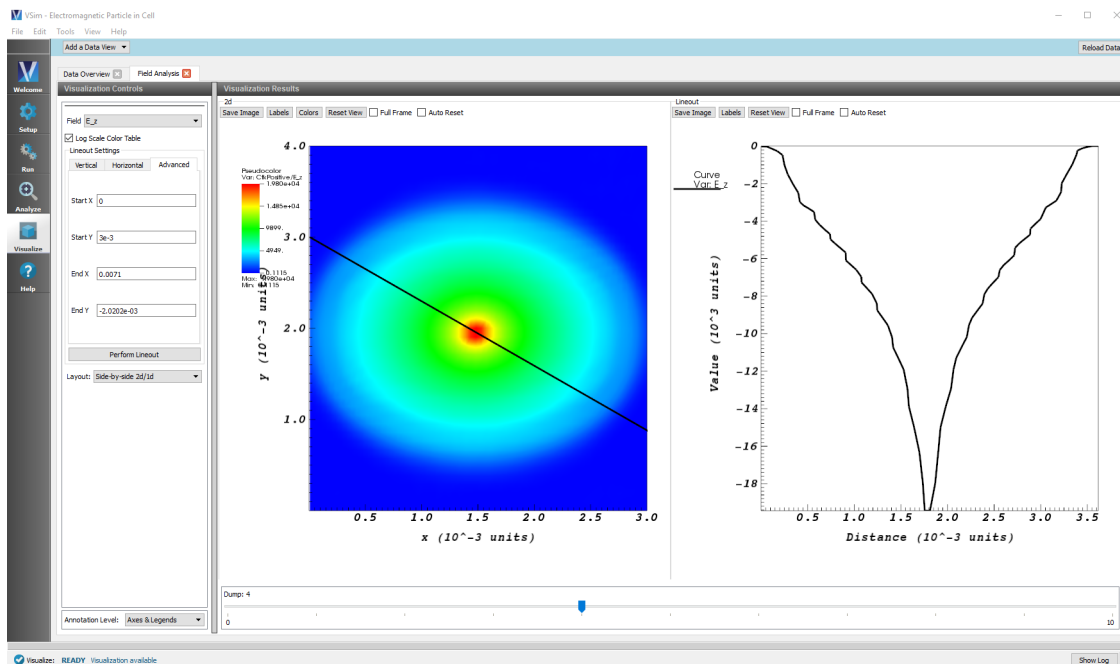


Fig. 14.18: The lineout settings controls

## Layout

The layout of the *Visualization Results* pane can be changed from the default of Side-by-side 2d/1d. Options include:

- Side-by-side 2d/1d
- Stacked 2d/1d
- 2d Only
- 1d Only

See [The layout dropdown options](#).

## 14.7 History

The *History* data view allows for plotting of any 1D array histories that were included in your input file prior to running your simulation. You can select this from the *Add Data View* drop-down menu at the top left-hand corner of the pane.

Up to 2 histories can be viewed at one time.

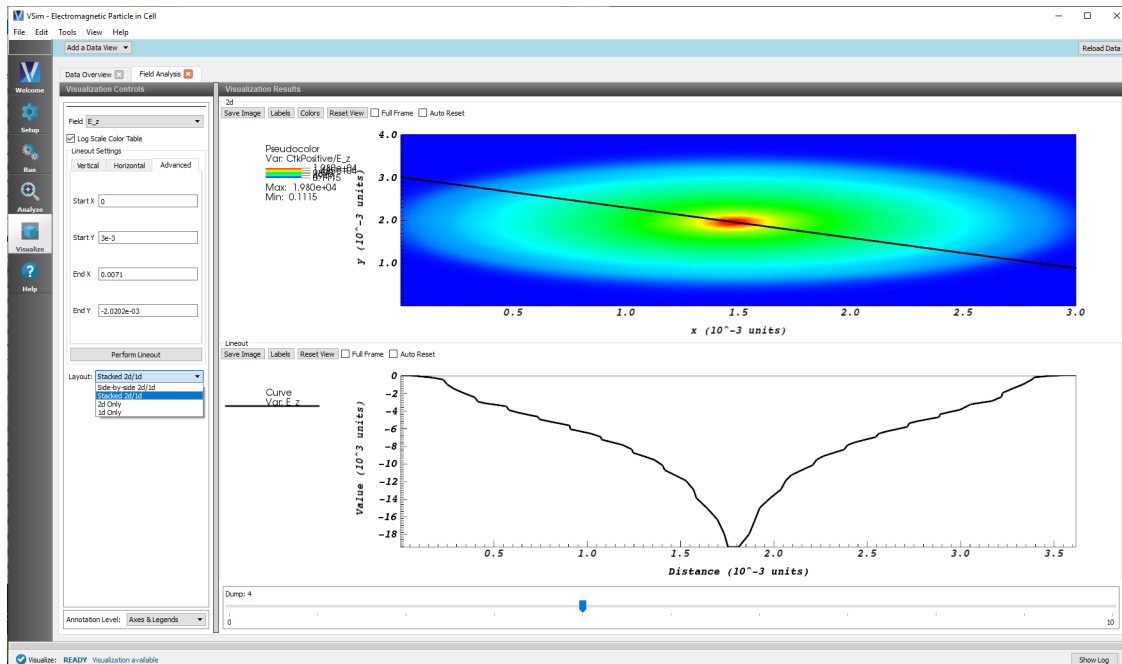


Fig. 14.19: The layout dropdown options

## Add Window

This tool can have two windows. This button will add second window if only one is present.

## Add Curve

The *Add Curve* button shows a dialog that allows you to plot multiple histories on top of each other. Choosing a variable from the drop down of the dialog and clicking ok will plot the variable on the same window.

## Edit Curve

The *Edit Curve* button shows a dialog that allows editing of the selected curve. The Dialog has the following options.

- *Select Curve*: Allows selection of the curve which is to be edited. Choose *All* if you want to edit all the plots in that window.
- *Color*: The color of the line can be modified using the *Color* drop down. Choices include red, blue, green, and black.
- *Style*: The style of the line can be modified using the *Style* drop down. Choices include solid, dash, dot, dotdash, and points.
- *Remove Selected Curve*: Removes the selected curve from the window. The window will be closed if all curved are removed. New window can be added from the *Add Window* button.

See [The Edit Curve Dialog](#).

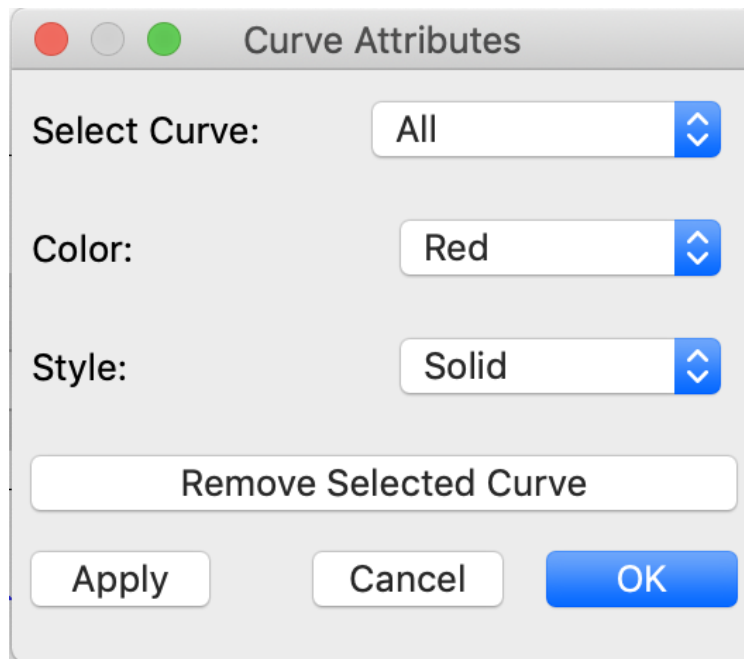


Fig. 14.20: The Edit Curve Dialog

### Fourier Amplitudes (dB)

Selecting the *Fourier Amplitudes* checkbox will take a Fast Fourier Transform of the data.

### Log of Absolute Value

Selecting the *Log of Absolute Value* checkbox will put the y-axis on a log scale.

### Zoom

Setting the *Zoom* radial selection will switch the mouse click feature to zoom. Start by clicking the mouse inside the plot window and then dragging to create a rectangle. Finish by un-clicking the mouse button. The plot will be zoomed to the data contained inside the rectangle.

### Navigate

Setting the *Navigate* radial selection will switch the mouse click feature to navigate. Click the mouse inside the plot window and drag to move the plot. See [The History View](#).

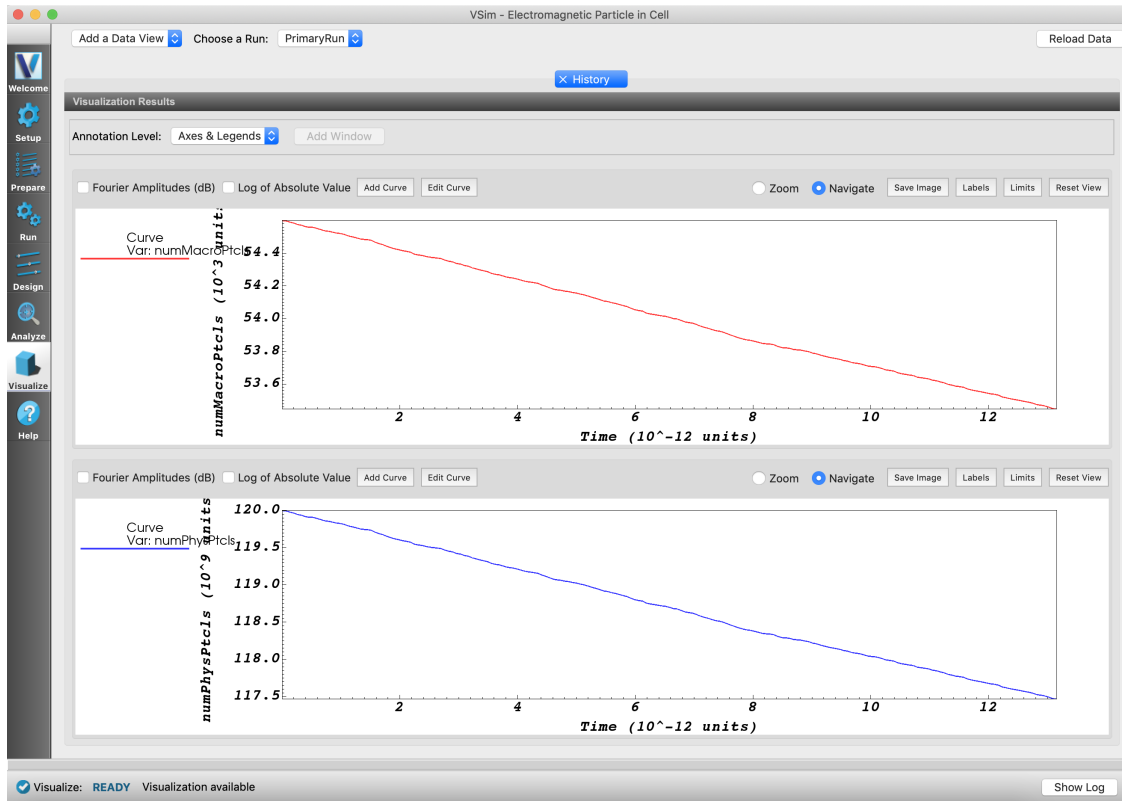


Fig. 14.21: The History View

## 14.8 Phase Space

The *Phase Space* data view allows for plotting of any particles (species) in your simulation. You can select this from the *Add Data View* drop-down menu at the top left-hand corner of the pane.

### Base Variable

The *Base Variable* can be used to switch between any of the particle species in your simulation.

### **X-axis**

The variable to be plotted on the x-axis.

### **Y-axis**

The variable to be plotted on the y-axis.

### **Z-axis**

The variable to be plotted on the z-axis.

### **Color**

The particles can be plotted in either a solid color such as red, green, or blue, or they can be plotted using another variable as their color. An example is to plot the velocity as the color on an x, y, z spatial plot.

### Point Size (pixels)

The size of each particle symbol.

### Symbol

The *Symbol* pulldown menu contains choices for the following shapes:

- Point (default)
- Box
- Axis
- Icosahedron
- Octahedron
- Tetrahedron
- Sphere Geometry
- Sphere

### Enable Second (Third) Plot

Up to 3 particle species can be plotted at one time in the *Phase Space* window. To enable another plot, check the box.

### Reset View on Draw

When changes are made to the variables to each plot, you must click the *Draw* button. Check the *Reset View on Draw* button if you would like the view reset each time the draw button is clicked.

### Draw

When changes are made to the variables to each plot, you must click the *Draw* button to redraw the plot.

See *The Phase Space View*.

## 14.9 Binning

The *Binning* data view allows for “binning” the particles (species) in your simulation and creating histogram-style plots. You can select this from the *Add Data View* drop-down menu at the top left-hand corner of the pane.

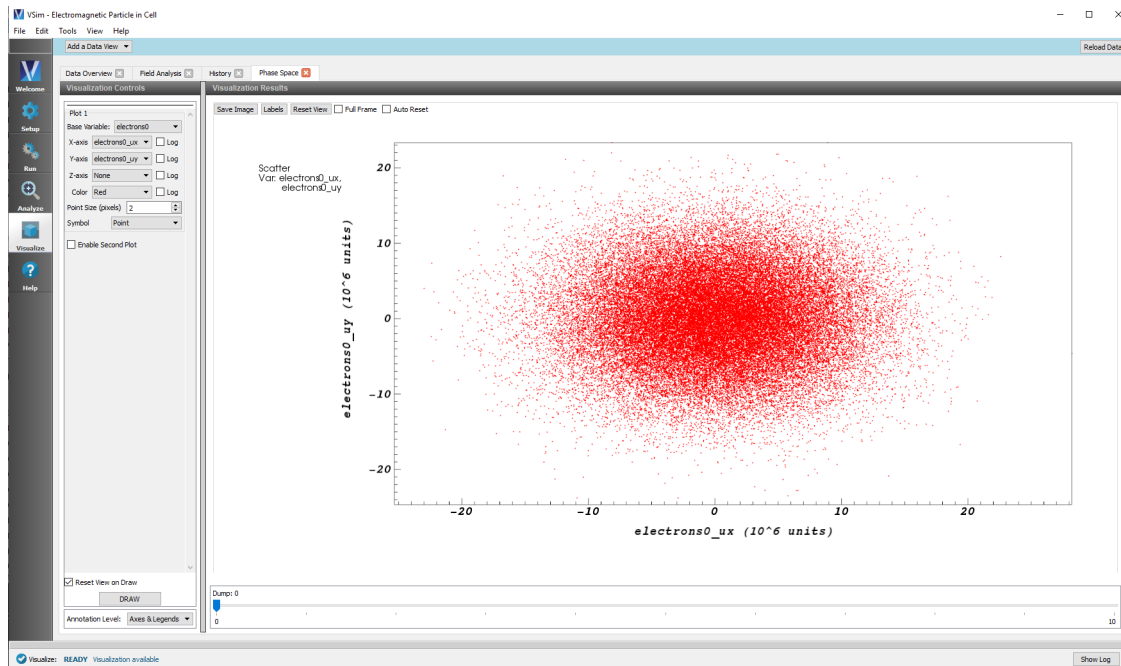


Fig. 14.22: The Phase Space View

### Base Variable

The *Base Variable* can be used to switch between any of the particle species in your simulation.

### Variable

The *Variable* to be binned.

### Bins

The number of *Bins*. The variable will be divided into the number of bins chosen.

### Reduction Operator

The *Operator* is the method used for binning.

### Reduction Variable

If the *Variable* is active (depending on the operator), the variable what the operator acts on.



## Draw

When changes are made to the variables, you must click the *Draw* button to redraw the plot.

## Slicing

Slice settings will appear when a third dimension is added. The slice settings allow you to set the position of the slice of the 3D field to create the 2D plot. The *Plane Controls* button allows for further control, including creating a slice at an angle. See *The Binning View*.

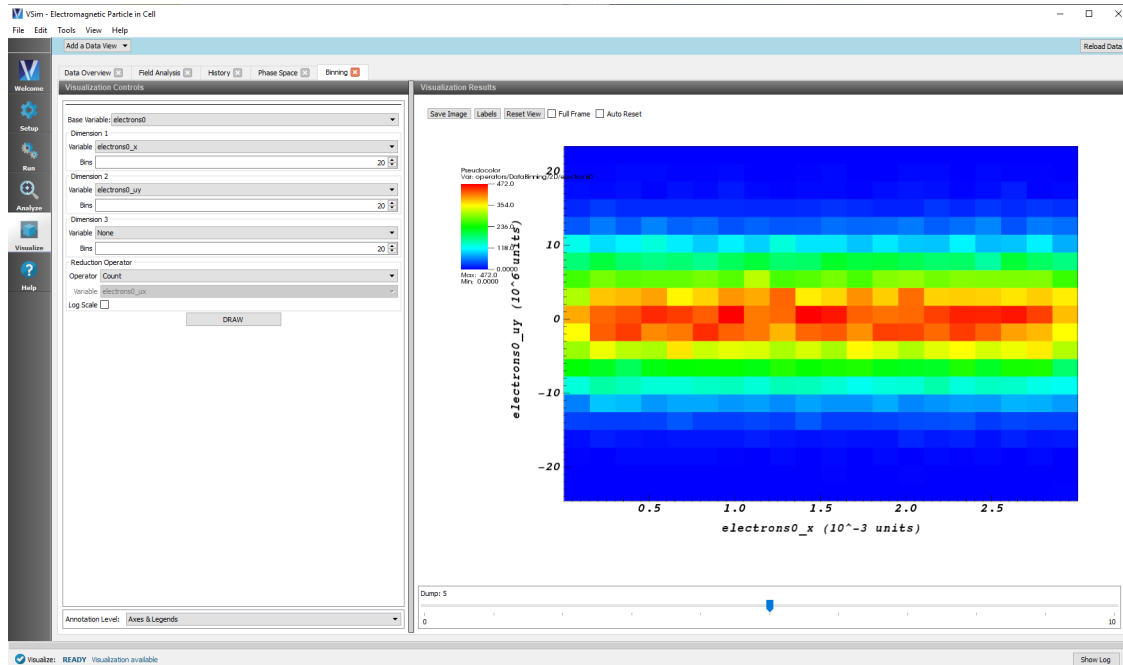


Fig. 14.23: The Binning View



## TROUBLESHOOTING

### 15.1 Troubleshooting Electrostatic Simulations

#### 15.1.1 Simulation Crashes at Initialization in Parallel

This is most commonly due to under or over specified boundary conditions on the simulation. It is common for the simulation to be able to run in serial but not parallel in this condition. Visual Setup simulations will take care of specification except in the event a partial boundary condition on one simulation face intersects a partial boundary condition of a separate face. In that event it is critical to make sure that the lower bound of the selected boundary surface is filled as well as up to one cell above the upper bound of the boundary surface, and that neither of these intersectins are set twice.

#### 15.1.2 Simulation Crashes at Startup with PEC Dirichlet Boundaries

PEC objects in the simulation must be entirely inside of the simulation grid in order to ensure proper problem setup.

#### 15.1.3 The Electrostatic Solver Does Not Converge

If the electrostatic solver does not converge, this often indicates a problem with the setup. The matrix can be singular in a fully periodic system due to the failure to specify the value of the potential at one point.

### 15.2 Troubleshooting Electromagnetic Simulations

#### 15.2.1 The Simulation Does Not Finish Properly

The most common cause of crashes is improperly set up particle boundaries. The particle boundaries must completely surround the space in which particles are loaded. Otherwise particles can drift out of the grid and try to reference fields that do not exist. This leads to a Vorpel segmentation fault.

## 15.2.2 The Output Shows an Unexpected High-Frequency or Checkerboard Pattern

A common problem with electromagnetic simulation is not following the Courant condition [CFL28]. The Courant condition states roughly that the time step must be small enough that a light wave cannot cross more than one cell in a single time step.

High-Frequency and checkerboard patterns can be symptoms of an instability resulting from violating the Courant condition.

## 15.3 Troubleshooting Visual Setup Crashes

The Visual Setup makes setting up simulations much simpler than writing an input file, but the output messages when a simulation crashes can be mysterious. If a simulation crashes before the engine takes the first step, it is more than likely that there is an error in the `.sdf` file. Documented below are some common errors, and the output messages they will create at runtime. If your simulation is crashing before the first step, take a look to see if any of the error messages shown below matches yours.

If your error message is NOT below, send an email to [support@txcorp.com](mailto:support@txcorp.com) and an Application Engineer will take a look at your simulation, and maybe your error will show up in the documentation for the next release!

### 15.3.1 General Tips

- Check the `.in` file for any unevaluated expressions.
- The message “Object %%%: Could not locate object \* in the input file object hierarchy” usually means that something is missing from the `.sdf` file. Either a variable used in the simulation isn’t present or is miss-named, or an attribute isn’t set/is left blank. Try to figure out what the “%%%” object is in the simulation then figure out what is missing. See below for some specific examples the examples.
- Running in parallel can sometimes suppress error messages. If you get a crash try running in serial for a better error message.
- If a simulation with particles makes it to the first step, then hangs, it is possible that more particles than GSim can handle are being loaded into the simulation.
- Objects need to be assigned a material before they can be used else where in the simulation.
- Geometry objects cannot set boundary conditions on the walls of a simulation. That is, a boundary condition with the “boundary surface” attribute set to “shape surface” will fail to set a boundary on the upper x, lower x, upper y, etc surfaces of the simulation.
- When using a geometry object to set a boundary condition internal to the simulation boundaries, make the geometry extend beyond the nodes on which you wish to set the boundary.

### 15.3.2 Parameter/Constant Not Named Properly

The following error was created in a simulation that had a constant named *BEAM\_RADIUS* but had a SpaceTimeFunction that used “BEAMRADIUS”.

Here, the unknown expression (“undefined symbol”) was printed out, providing an avenue for beginning the debugging process.

```

Sources are:
  BeamEmitter
  sputterNeutralEmitter0

ERROR: In setting up simulation:
TxSymbolTable::checkUndefVar: Undefined symbol 'BEAMRADIUS'.

Settings in the Z axis will be ignored
Lines from 'ionBeamSputtering.pre' processed.
Finished with 'ionBeamSputtering.pre'.
Error building data.

----- END ENGINE OUTPUT -----

Engine completed with error: VORPAL INPUT-FILE ERROR (code 5)

```

### 15.3.3 History Attribute Not Set

This error was from the same simulation as the error above. A history to count the number of physical particles was added, but the species was left blank.

Unlike the error above, there is no “ERROR: ...” statement printed out. The nature of the error has been highlighted in the image below for visibility. Sometimes, the nature of the error can be buried in the output, so keep a sharp eye out for statements like the one below, because it does (indirectly) point to the cause of the crash.

```

Source named BeamEmitter added to Species ArgonIons.
Source named sputterNeutralEmitter added to Species ArgonIons.
Building species neutralCopper with NDIM = 2.
Added sink, lower0Absorber, to global region:
[-1, 0) x [-1, 33)Added sink, upper0Absorber, to global region:
[32, 33) x [-1, 33)Added sink, lower1Absorber, to global region:
[0, 32) x [-1, 0)Added sink, upper1Absorber, to global region:
[0, 32) x [32, 33)ERROR: In setting up simulation:
Object numPhysPtcIs0: Could not locate object please in the input file object hierarchy..

Settings in the Z axis will be ignored
Lines from 'ionBeamSputtering.pre' processed.
Finished with 'ionBeamSputtering.pre'.
Error building solvers.

----- END ENGINE OUTPUT -----

```

### 15.3.4 Missing Attribute in Boundary Condition

This error is very similar to the one above (History Attribute Not Set). In this case a Dirichlet boundary condition named “dirichlet1” was added to the simulation, but a surface for the boundary condition was not specified.

The name of the boundary condition that was added was “dirichlet1”. In the image below, notice that this name appears in the error message (“Object dirichlet1Filler ...”).

```

ERROR: In setting up simulation:
Problem found setting up MultiField:
Object dirichlet1Filler: Could not locate object selectSurface in the input file object hierarchy..

Settings in the Z axis will be ignored
Lines from 'ionBeamSputtering.pre' processed.
Finished with 'ionBeamSputtering.pre'.
Error building solvers.

----- END ENGINE OUTPUT -----

```

### 15.3.5 Over-Specifying Boundary Conditions

This error arose when two, overlapping CSG geometry objects were used to set the voltage on a region in the simulation, so the voltage was over-specified.

Depending on which electrostatic solver is being used, a different message will be output. The first error message below was the result when using the “superLU” direct solver:

```
Lines from 'ionBeamSputtering.pre' processed.
Finished with 'ionBeamSputtering.pre'.
ERROR: An unhandled exception was raised: Error! The parameter "Total numeric factorization time" does not exist
in the parameter (sub)list "ANONYMOUS".

The current parameters set in (sub)list "ANONYMOUS" are:

{
  "Total matrix redistribution time" : double = 0
  "Total Amesos overhead time" : double = 0
  "Total symbolic factorization time" : double = 0
}

Throw number = 1

----- END ENGINE OUTPUT -----
```

This message was output when using the “gmres” iterative solver:

```
Species neutralCopper has velocity limits = [929072, 464536].
Sinks are: lower0Absorber upper0Absorber lower1Absorber upper1Absorber
-----
Using 'power-method' for eigen-computations
***
*** ML_Epetra::MultiLevelPreconditioner
***
Matrix has 1103 rows and 4699 nonzeros, distributed over 1 process(es)
The linear system matrix is an Epetra_CrsMatrix
** Transforming column map of Main linear system matrix
Default values for 'DD'
Maximum number of levels = 10
Using increasing levels. Finest level = 0, coarsest level = 9
Number of applications of the ML cycle = 1
Number of PDE equations = 1
Aggregation threshold = 0
Max coarse size = 128
R and P smoothing : P = (I-\omega A) P_t, R = P^T
R and P smoothing : \omega = 1.333/lambda_max
Null space type = default (constants)
Null space dimension = 1
----- END ENGINE OUTPUT -----
```

### 15.3.6 Under-Specifying Boundary Conditions

This is an example of an error where the printed error statement provided little to no help in debugging, unless you are deeply familiar with matrix solver packages (Epetra constructs matrices and is the part of the Trilinos Library which is used by GSim).

The cause of this error is that there are insufficient boundary conditions resulting in an incomplete matrix. If you see this error, adjust your boundary conditions to make sure boundary conditions are completely set on all boundaries of the simulation. Note: geometries cannot be used to set boundary conditions on the walls/boundaries (lower x, upper x, lower y, etc.) of a simulation.

```

ERROR: In setting up simulation:
Problem found setting up MultiField:
Epetra error -1 occurred calling FillComplete() on matrix..

Settings in the Z axis will be ignored
Lines from 'ionBeamSputtering.pre' processed.
Finished with 'ionBeamSputtering.pre'.
Error building solvers.

----- END ENGINE OUTPUT -----

```

### 15.3.7 Particle Loader Missing Loading Volume

Below is an example of what happens when a particle loader is not setup correctly. This is another more mysterious error message, since after the “ERROR: In Setting up simulation” there is very little to indicate the location of the bug. However, notice that the last thing printed before the error was “Sources are: /n particleLoader0”. This is a hint as to where to check your input file.

```

Sources are:
particleLoader0

ERROR: In setting up simulation:
No prmVec named 'lowerBounds'.

Settings in the Z axis will be ignored
Lines from 'ionBeamSputtering.pre' processed.
Finished with 'ionBeamSputtering.pre'.
Error building data.

----- END ENGINE OUTPUT -----

```

### 15.3.8 Fluid Missing Volume

This error is almost identical to the particle loader error listed above. A background fluid was added to the simulation, and the volume attribute was not set. Notice how the simulation crashes right after printing out a statement about the “neutralFluid1”.

```

[0, 32) x [-1, 0)Added sink, upper1Absorber, to global region:
[0, 32) x [32, 33)neutralFluid1 has the grid initial conditions: initialDensity.
ERROR: In setting up simulation:
No prmVec named 'lowerBounds'.

Settings in the Z axis will be ignored
Lines from 'ionBeamSputtering.pre' processed.
Finished with 'ionBeamSputtering.pre'.
Error building solvers.

----- END ENGINE OUTPUT -----

```

### 15.3.9 Unable to Open SDF (No Such Attribute Error)

If after creating a simulation with an imported CAD file and changing the file import, it cannot reopen with an error “getAttrib const: No such attribute. Attributes are...” This can be caused by a CAD file with multiple parts having the same name, and then the number of parts with multiple names changing. For example if a single .step file is imported twice, and then a single deleted. This is due to an issue with how GSim handles parts with multiple identical names. It is not an issue if the number of imported CAD file’s stays consistent.

To correct this issue contact Tech-X Support.

### 15.3.10 Geometries Having Poor Lighting

If your simulation uses geometries or grids smaller than 1mm or larger than 100m, click the ‘Show Scale’ button in the 3D view and use the scale slider to select the appropriate scale for rendering. This will prevent issues with geometry visibility, axis labeling, and item coloring.

### 15.3.11 Use of Restricted Variable Names (selective processing only)

The following variable names cannot be used when selectively processing files. This is not an issue in standard use of GSim/Visual setup.

ENDMARKER, NAME, NUMBER, STRING, NEWLINE, INDENT , DEDENT , LPAR , RPAR, LSQB RSQB, COLON, COMMA, SEMI , PLUS , MINUS, STAR , SLASH, VBAR, AMPER, LESS, GREATER EQUAL, DOT, PERCENT, BACKQUOT, LBRACE, RBRACE, EQEQUAL , NOTEQUAL, LESSEQUAL GREATEREQUAL, TILDE, CIRCUMFLEX, LEFTSHIFT, RIGHTSHIFT, DOUBLESTAR, PLUSEQUAL MINEQUAL, STAREQUAL, SLASHEQUAL, PERCENTEQUAL, AMPEREQUAL, VBAREQUAL, CIRCUMFLEXEQUAL LEFTSHIFTEQUAL, RIGHTSHIFTEQUAL, DOUBLESTAREQUAL, DOUBLES LASH, DOUBLES LASH EQUAL, AT ATEQUAL, OP, COMMENT, NL, RARROW, ERRORTOKEN, N\_TOKENS, NT\_OFFSET

## 15.4 Troubleshooting Plasma Density

If *applyTimes* is set, and includes time  $t=0$  in the range, particles may be loaded both during initialization and in the zeroth time step of the model, potentially giving twice the density you want.

Use together with a history of *kind=speciesNumPhysical* and compare the total population to the number of particles you expected.

Use together with the *computePtclNumDensity* script to check the overall density of the loaded particles.

## 15.5 Troubleshooting Missing Secondary Particles

If *suppressEnergy* is left to its default value of 0 in text driven simulations, this will prevent loading of secondary electrons subject to an E field which would accelerate it into the boundary. The user can try switching off the sources of E fields to check if the emission is correct. Due to space-charge it's quite likely that many of the secondary electrons will then experience the field and be driven into the boundary, having their emission suppressed. In other words they will not appear in the simulation. Setting *suppressEnergy* to a large value like  $1.e20$  should address the issue.

See also the descriptions of Secondary Electron Emitter and *secElec* in the Reference manual.

## 15.6 Troubleshooting Crashes During Stepping of Particle Simulations

Once the simulation starts timestepping, most potential causes of premature termination of the simulation have been passed. If the simulation does end prematurely, there are several steps we can take to diagnose what is happening and speed up resolving whatever issues are causing these.

Firstly, there are several potential causes for crashes:

- The simulation is taking up too much memory.
- The simulation is trying to access areas of memory it should not.



- The simulation is looking for an object which it cannot find.
- The hard drive has run out of space.

If the simulation is taking up too much memory, it is a good idea to determine if the number of particles in the simulation is rising steadily or running away. Instrumenting a history to measure the number of macroparticles in each species is a good way to determine if this is the cause. Rerun the simulation to a few steps short of where the crash happened previously, then reduce the `dumpPeriodicity` to get more information about the steps where the crash is occurring. If the crash is due to a runaway of the number of macros inside the simulation volume, then it will often be preceded by a slowing down of the wall-clock time per step.

If the number of particles is reasonable, then it is possible for particles to be finding their way out of the simulation domain. All particle simulations should include an absorbing box on the perimeter of the simulation. This is automatic in visual setup simulations, and is the responsibility of the user in text setup examples. In some rare cases there is an interaction between a separate absorber on a geometry and the perimeter absorber that can lead to crashes due to particles getting outside the domain. The simulation is trying to determine a field at a location that is not in memory. This is very rare and should be reported to [support@txcorp.com](mailto:support@txcorp.com). This can also happen if the particles are being accelerated out of the domain. In these cases, the increase of dump periodicity is useful, but so also is the `computePtcLimits` analyzer, which will find particles at highest x, lowest x, highest velocity, and report that information in a history. If this shows particles exiting the simulation, then the input file(s) can be adjusted to avoid the possibility that the particles might escape there.

Other situations where a crash may occur later in a simulation include, where a secondary emitter is poorly defined, and it is some time before the shape on which that emission occurs is struck by a particle that can trigger this process.

## 15.7 Troubleshooting Performance

GSim is designed to optimally use the computational hardware you have available, whether a laptop or a leadership class supercomputing facility.

It achieves this through the use of advanced algorithms, but this does not guarantee any given simulation will run as fast as possible. This document outlines some simple checks which may aid speeding up a simulation.

On the one hand, there are different types of algorithms for field solves, particle movers and monte-carlo, and these may all scale up in slightly different ways, and require different amounts of inter-processor communication for large parallel simulations. Having many histories may also impact performance.

Firstly, it helps to understand which parts of the simulation are taking the most time. The best way to do this is to remove elements of the simulation one at a time, and to assess the difference in speed. Having measured performance of field solves, particle pushes (if applicable), monte carlo interactions (if applicable) and history objects, it may be possible to simplify some of these, for example by adjusting the number of physical particles per macroparticle.

In general one need not dump the fields and particles more often than is necessary as this will lead to slow visualisation, and the slowing down of the simulation while the data is written.

### 15.7.1 Electromagnetic Solves

Electromagnetic solves tend to be bound by memory access and the ability to pass boundary data across the network. As a rule of thumb - performance tends not to increase well when the domain on each processor is smaller than 40x40x40 - but this limit will depend on the relative performance of your network fabric and CPU. Also, cells outside perfect electrical conductor take longer than inside, so it can sometimes be worth adjusting the domain decomposition strategy to ensure the load is balanced equally. Minimize the regions over which any MAL or PML boundary conditions are applied, as these will be comparatively slow compared with a normal cell update.

## 15.7.2 Histories

Histories store their data in RAM in between data dumps and can write very large datasets. Some histories need to do non-trivial amounts of computation each time step.

## 15.7.3 Configuration Issues

The installer Tech-X provides can be expected to work well out of the box on desktop and high performance computing systems.

HPC systems often have high performance parallel systems. Commonly these are set up differently from your home area, and you will need to ensure that you are running with your data being output to a specific partition. Check the cluster documentation for more information.

HPC systems are sometimes configured with a different MPI (to GSim's required MPICH MPI) in the environment set up by the queue system. In rare cases the MPI installation provided by GSim can pick up the wrong network card or fail to use the correct infiniband driver (normally where this has been customized heavily on those clusters). This will likely manifest as very poor parallel performance. For example, a simulation using sixteen cores and one node may run much faster than a simulation on thirty-two cores and two nodes (subject to the scaling advice above). In these cases we recommend you contact Tech-X support at [support@txcorp.com](mailto:support@txcorp.com) for advice. Modification of the environment is non-trivial and may have unexpected consequences.

## 15.8 Troubleshooting MPI failure to start on OSX

On some versions of OSX, users have reported issues with MPI processes failing to start. This problem can be identified as it causes errors of the kind

```
[[20648,0],0] bind() failed on error Address already in use
```

It appears a solution for this is simply to run:

```
export TMPDIR=/tmp
```

Then to run GSimComposer from the same terminal.

Another occasional problem shows up with the following error:

```
Fatal error in internal_Init: Other MPI error, error stack:
```

Followed by lengthy error output.

One solution is to add the following to /etc/hosts or /private/etc/hosts :

```
127.0.0.1 yourMachineName
```

Where yourMachineName is the name of your machine

## 15.9 Troubleshooting Windows Permissions

If you see permission errors, try running the command prompt as administrator by right clicking the prompt and clicking “Run as administrator”.

## 15.10 Troubleshooting GSimComposer Visualization

### 15.10.1 On Windows, GSimComposer Claims No Data Found Even Though it Exists

It is possible you are trying to visualize data from a folder other than a straight forward hard drive on the system. VSim relies on VisIt for visualisation and the following are known to cause problems:

- Simulation directory path on a remote file system
- Simulation path containing international characters
- Simulation path uses the Universal Naming Convention (starts \\)

On Windows machines, data inside a directory following Universal Naming Convention (UNC), e.g., \\machine\directory\file, cannot be visualized using VisIt, the underlying visualization tool to GSimComposer. UNC style paths are not supported. Instead, map the UNC path to a letter drive on the machine.

### 15.10.2 Field Plots Not Being Updated

In certain simulations, you may be plotting multiple fields and notice that certain ones advance in time while others never update past time 0.

This is due to the fact that the latter fields, as illustrated by the J field in the *Magnetic Fields of Wire (Text-based setup)* example simulation, are actually static fields that are never updated throughout the simulation run. Thus, when these static fields are examined in the *Visualization* window, only the initial field data at time 0 is displayed because that is the only data generated.



## GLOSSARY

**domain**

The rectangular Cartesian grid. The physical domain is the grid specified by a user. The extended domain is the grid with guard cells added by Vorpal.

**extended domain**

See domain.

**FDTD**

Finite-difference time-domain. The FDTD method is a technique for solving problems in electromagnetics.

**float**

A floating-point number.

**guard cell**

A cell located outside the user-defined simulation grid that Vorpal adds for parallel processing and other computational purposes. Charges cannot be deposited in guard cells, but you can use guard cells when you describe boundary conditions.

**HDF5**

Hierarchical Data Format Version 5. A library and file format, developed by the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign, for storing graphical and numerical data and for transferring that data between computers. Vorpal and VSimComposer output data in hdf5.

**input block**

An input block is an object consisting of parameters. Input blocks can be nested within other input blocks. For example, input blocks for boundary conditions are nested within the input block for an electromagnetic field.

**input file**

A Vorpal simulation file, which has a .pre suffix. Users define a simulation and its variables in an input file. VSimComposer then runs the input file through a preprocessor to produce a processed input file.

**MPI**

Message Passing Interface. An application programming interface (API) for communicating between processes executing in parallel.

**multi-grid pre-conditioner**

A pre-conditioner that enables a solver to use a hierarchy of grids to solve a partial differential equation problem. The multi-grid pre-conditioner applies the results from coarse grids to accelerate the convergence on the finest grid.

**parameter**

A parameter is a variable value (integer, floating-point number, or text string) that users define to create a simulation.

**parse**

To divide input into parts and determine the meaning of each part.

**physical domain**

See domain.

**pre-conditioner**

An algorithm that works with an electrostatic solver to transfer an original linear system matrix into a matrix that has better convergence behavior.

**processed input file**

A Vorpak simulation file, which has a .in suffix. VSimComposer processes the input file to produce a processed input file.

**Python**

An open-source, interpreted scripting language managed by the Python Software Foundation.

**SI units**

The International System of Units (*Le Systeme International d'Unites*), which has seven base units: meter, kilogram, second, ampere, kelvin, mole, and candela.

**solver**

An algorithm that calculates the results of electrostatic problems.

**TxPhysics**

A cross-platform library of computational modules, provided by Tech-X Corporation, for modeling charged particles.