PLASIMO User Guide

The PLASIMO Team

July 3, 2020
## Contents

1 Introduction 4

2 Downloading, Installing and Running PLASIMO 5

2.1 Downloading PLASIMO . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
2.2 Starting PLASIMO . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
2.3 Installing and Running a Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

3 The Graphical User Interface 9

3.1 The Main Window . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
3.2 The Global Settings Window . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
3.3 The Model Editor Window . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
  3.3.1 The Tree Editor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
  3.3.2 The Leaf Editors . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
3.4 The Data Window . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
  3.4.1 Additional Functionalities of the Viewers . . . . . . . . . . . . . . . . . . . . . . . . 14
  3.4.2 Initial Viewers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
3.5 Model-specific GUI Extensions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

4 Custom expressions in PLASIMO 17

4.1 Custom expressions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
4.2 Supported Operators and Functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
  4.2.1 Variables that can be used in expressions . . . . . . . . . . . . . . . . . . . . . . . . 17

5 PLASIMO Example Models 22

5.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
5.2 Drift-Diffusion Models . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
  5.2.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
  5.2.2 DC Glow Discharge Tube . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
  5.2.3 Sputtering Hollow Cathode Discharge . . . . . . . . . . . . . . . . . . . . . . . . . 26
  5.2.4 Building a 1D Drift-Diffusion Model from Scratch . . . . . . . . . . . . . . . . . . 27
  5.2.5 Building a 2D Drift-Diffusion Model Using BOLSIG+ . . . . . . . . . . . . . . . . . 30
5.3 Global Models . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
  5.3.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
5.3.2 Example Global Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
5.3.3 Building a Global Model from Scratch . . . . . . . . . . . . . . . . . . . . . . . . . 36
      5.3.4 Solver Issues . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
5.4 Heat Transport Models . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
  5.4.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
  5.4.2 Transient Heat Conduction in a One-dimensional Medium . . . . . . . . . . . . . . 38
5.5 Non-LTE Models . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
  5.5.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
5.5.2 Cascaded Arc Model ................................................................. 40
5.5.3 Building a 1D non-LTE Quasi-neutral Model from Scratch .............. 41
5.6 LTE properties calculator ............................................................... 44
  5.6.1 Introduction ........................................................................... 44
  5.6.2 LTE properties of Ar mixture ...................................................... 44
5.7 Other Models ............................................................................... 45
  5.7.1 Monte Carlo model ................................................................. 45
  5.7.2 EM model - 1 ............................................................... 46
  5.7.3 EM model - 2 ............................................................... 47
Chapter 1

Introduction

PLASIMO is a framework for simulating low-temperature plasmas and gas discharges. It is being developed in the group Elementary Processes in Gas Discharges (EPG) at the Department of Applied Physics of Eindhoven University of Technology (TU/e). Since 2015 the TU/e's spin-off company Plasma Matters B.V. is responsible for the distribution of the software, providing software support and consultancy services.

For requesting an access to PLASIMO, please fill in the access request form on the PLASIMO web page [https://plasimo.phys.tue.nl/form_access_request.php](https://plasimo.phys.tue.nl/form_access_request.php) or contact the PLASIMO team info@plasma-matters.com.

For an overview of PLASIMO’s present capabilities we refer to the review paper [1] and to the PLASIMO website [https://plasimo.phys.tue.nl](https://plasimo.phys.tue.nl).

If PLASIMO is suitable for your particular application, you need to sign the software license. PLASIMO team will provide you with information about the terms and conditions. After obtaining a license you will be able to download the software and you will have access to all relevant documentation on this website.

This guide

This guide starts with a quick install procedure for PLASIMO. After this we will explain in more detail an important part of PLASIMO: its graphical user interface (GUI). Next, using the GUI we will install and run a series of example models to show the capabilities of our simulation toolkit.
Chapter 2

Downloading, Installing and Running PLASIMO

PLASIMO is available in two flavors: stable, released versions, and an experimental developer version, where all the changes are done for the next release.

Before downloading PLASIMO, you should take a decision which branch you want to use: “Stable” or “Developer” (also called HEAD). “Developer” is recommended for people who need the latest features of PLASIMO. The downside is that experimental changes are allowed to go in and carry the risk of breaking user input files or crashing the user interface. Users that need to update their input files are advised how to do so. The file NEWS contains the documentation on what has changed since the last release. Note that it’s possible to have more than one copy of the PLASIMO source tree on your machine, so you can work with several versions if needed.

2.1 Downloading PLASIMO

Assumes that you already have the rights for downloading PLASIMO and you have decided which version of PLASIMO you want to use. The PLASIMO packages can be downloaded from the PLASIMO’s web page [https://plasimo.phys.tue.nl/download.php](https://plasimo.phys.tue.nl/download.php).

Depending on your operating system proceed to the relevant section below.

For Microsoft Windows users:

Download the compressed zip file for Windows of the chosen version of PLASIMO. Extract the files on your hard drive. A typical installation directory may be `C:\Users\jan\`. The zip archive puts all files in a subdirectory with the name `plasimo-XXX`, so you end up with a directory `C:\Users\jan\plasimo-XXX`. This directory will be called the PLASIMO root directory hereafter.

Note that you do not need to have administrator rights to install the modeling package, as long as you have write access to the installation directory. In particular, installing PLASIMO does not require changes to the Windows Registry.

For Unix/Linux users: openSUSE

You can use the default Installer that will guide you in the installation procedure or you can save the rpm file and execute the command below after replacing the 'XXX' with the correct rpm file name:

```
zypper install plasimo-XXX-XXX.rpm
```

In both cases root privileges are needed to install PLASIMO. The installation directory is `/opt`. 
For macOS

The installation can simply be started by double-clicking the .mpkg file. When installing the package you might get an error message informing you that it originates from an “unidentified developer”. To install the package anyway, open “System Preferences” and go to “Security & Privacy”. If there is a message saying something similar to “plasimo.mpkg was blocked from opening because it is not from an identified developer.” you can press the button “Open Anyway”. Otherwise, temporarily “allow apps downloaded from: Anywhere” and try installing the package again.

2.2 Starting PLASIMO

PLASIMO can be started in two ways: with a graphical user interface (GUI) and with a non-interactive console application. Using the console application from the command line can be used when you can’t or don’t want to use the GUI. Reasons may be that wxWidget is not available on your platform or that you prefer to use the models non- interactively (batch processing) or that you want to save the (albeit small) performance hit from running a GUI.

Execution on Windows:

In order to facilitate starting the executables under an MS Windows environment, a set of batch files is shipped with the distribution, these can be found in the PLASIMO root directory.

You can start PLASIMO with the graphical user-interface by simply executing the script Plasimo.bat, which resides in the PLASIMO root directory, for example with Windows Explorer. Together with two console windows the PLASIMO main window appears. The second console window shows the log of PLASIMO. After starting the application with an ‘empty’ model, you should see a window like that in figure 2.1.

Execution on Linux/Unix:

As part of the build process, some information about the location of various files is recorded and stored in a scripts, set-plasimo. This will set the current working directory as /home/jan.

Execute the following commands from your home directory:

1. /opt/plasimo/bin/set-plasimo
   or source /opt/plasimo/bin/set-plasimo

2. /opt/plasimo/bin/wxplasimo

Before continuing with loading a model, you have to prepare a directory where PLASIMO can write the output data. You have to create a directory data in your home directory:

    mkdir data

After starting the application with an ‘empty’ model, you should see a window similar to the one shown in figure 2.1.

Execution on macOS:

Since macOS is certified as Unix, it can be run the same way as the Linux/Unix version. On a standard install of macOS the /opt directory where PLASIMO is installed is not shown in Finder, so you can not simply use the file selection dialog to open one of the demosuite files that are shipped with PLASIMO. The first option is to make the /opt directory visible in Finder with the following command:

    sudo SetFile -a v /opt/.

The second option is to use the “go to” feature of Finder by using the keyboard shortcut “Command + Shift + g” and typing /opt.

Execute the following commands from your home directory:
1. `/opt/plasimo/bin/set-plasimo`
   or `source /opt/plasimo/bin/set-plasimo`

2. `/opt/plasimo/bin/wxplasimo`

After starting the application with an ‘empty’ model, you should see a window similar to the one shown in figure 2.1.

![PLASIMO graphical user interface and Help window](image)

Figure 2.1: Two screen-shot of PLASIMO’s graphical user interface (GUI). The screen-shot of the main window (left figure) has been taken just after starting the application, when no model file has been loaded yet. The elements of the main window (left figure) are a menu bar, a tool button bar and set of tab windows. In the figure a tab window called ‘model editor’ is active. Context-sensitive help text is displayed in a second window (right).

### 2.3 Installing and Running a Model

In the application main window you see a menu. In order to allow quick access to the menu items, most commands are also available through the toolbar. Initially, the buttons **New**, **Open**, **Reload**, **Save**, **Help** and **Install** are active. While the icons are easily recognized, in case of doubt you may move the mouse over a button to see what it does: after a short time a ‘tooltip’ will be displayed. The remainder of the main window displays the properties of the active model. We will get back to that later.

1. **New** button gives you the possibility to create a new model by selecting a model type from the pop-up menu.

2. **Open** pops up a file dialog which allows you to load a model by selecting an existing input file. A number of existing models are shipped with the PLASIMO distribution. All the PLASIMO input files reside in the directory **input/**. The location of this directory depends on the specific PLASIMO version you are running:
   - Windows zip: the installation directory;
   - Linux package: in directory `/opt/plasimo-XXX-XXX` (or appropriate softlink `/opt/plasimo`);

The files in the **input/** directory are separated in sub-folders depending on the sub-modules of PLASIMO. Some of the existing example input files are described in more detail in the chapter 5.
3. Please load the example file `plplugins/flow_temperature/input/cylindrical_2d_atmospheric_pressure.gum`.
If there were no errors, the main window should now look like Figure 2.2. In the tree window on the left you see a tree representation of the model. The window on the right shows the contents of the active tree item. These editors will be discussed in detail in chapter 3. Tutorial sessions are provided in chapter 5.

4. The **Reload** button allows you to reload an input file. PLASIMO uses simply text files that can be edited with any of your favorite editors. In case you made changes in the this text input file, this provides a quick way to reload the modified model.

5. The **Help** button activates or deactivates the visibility of a help window. By default it is activated. It shows you a detailed description and requirements of the active tree item.

6. After a model has been loaded or modified to the user’s needs, it must be prepared for running. Press the **Install** button to do this. This will do all kinds of necessary one-time installations, and perform various checks on the sanity of the model file. Installing a model might therefore take some time, typically up to a few seconds. If anything goes wrong during installation, a message box will be popped up, telling the user what went wrong. After the problems have been solved, one can try to install the model once more.

7. When the model is successfully installed, additional buttons are activated. These are:
   - **Run**. Run the successfully installed model.
   - **Proceed (one step)**. All models are assumed to run in steps. A step can be either a time step, or an iteration. Selecting this item causes a (paused) model to do one such step. This can be useful if you use the application interactively, for example to find out the reasons for divergence of otherwise unexpected behavior.
   - **Stop**. Pauses a running model; this can be useful to inspect intermediate results.

For more information how to modify an existing model and how to manipulate the input data, please follow the steps in 5. Information about all the functionalities if the graphical user interface of PLASIMO, please read chapter 3.
Chapter 3
The Graphical User Interface

This chapter discusses the usage of the graphical user interface (GUI), its functionalities such as the various model editors, settings and model specific GUI extensions.

3.1 The Main Window

After starting the application you should see a window like that in figure 2.1(a). This application main window contains a number of elements, from top to bottom we have:

- A menu bar, containing the menus File, Run, Help
- A tool bar, discussed in section 2.3
- A set of tab windows, named Global settings and Model editor
- A status bar, displaying the current status of the model Installed, Paused, Running, etc.

As the menu/tool bars were previously discussed in chapter 2.3, the following text provides detailed information about the tab windows Global settings, Model editor and the additional Model data window, that appears after installing a model, as well as some model specific extensions.

3.2 The Global Settings Window

In the graphical user interface the contents of the global configuration file are visualized in the tab window Global Settings. It is important to notice that some of the settings in the configuration file take effect only after the application is closed and restarted, while others take effect while it is still running. Among other things this can be used to fine-tune the behavior of PLASIMO (changing the appearance of plot data and the like).

In the interface the active section is explained in the HTML help window. For an overview of all available sections, we refer to the PLASIMO user reference guide.

Most people can use PLASIMO without ever needing to use this window. Therefore, we shall continue this text with a discussion of the model editors and how you can edit a model using the GUI.

3.3 The Model Editor Window

Initially, the Model editor window has been activated. It is one of the tab windows previously mentioned and allows the user to create new model or edit existing one. Only one PLASIMO model can be active, multiple documents are not supported.
In general, the structure of the **Model editor** window resembles the interface of most file managers, showing the directory (section) tree on the left (**Tree editor**), while the window on the right shows the contents of the active section (**Leaf editor**).

### 3.3.1 The Tree Editor

The tree view shows the *nodes* or *sections* which are at the basis of the hierarchic structure of PLASIMO input files. Please note that the structure is *recursive*, that is, sections may contain (sub)sections themselves. Each section contains a group of conceptually related subsections and variables.

A section can be activated by clicking the left mouse button on the section entry in the tree. The variables which belong to the active subsection are shown in the ‘section window’ at the right hand side. Furthermore, in the **Help** window a detailed description of this section is displayed.

If you want to add a subsection, click the right mouse button on the section to which you want to add it. This will pop-up a context sensitive menu which allows you to make an appropriate selection. This way it is guaranteed that you add subsections only at locations where the model will actually look for them. By clicking on the check box with the right mouse button, a pop-up menu appears which offers the following choices:

- **Cut** Removes the active node and the contents is copy to the clipboard.
- **Copy** Copy the active node
- **Paste** Paste the copied/cut node
- **Paste into section** Paste a copied node into the active section
- **Disable** this disables the entry if it is enabled. Disabled sections can be recognized by the light-up color
- **Enable** this enables the entry if it is disabled. Pressing the right mouse button over a disabled section pops up a menu which allows to re-enable the section.
- **Undo all changes** This option will undo the results of editing the active node. Node that after switching to another node, editing can no longer be undone.
- **Add...** Add a relevant context dependent section. It is possible to have multiple choices. After one is selected a small addition window appears with the following options:
  - **Create default section** This will create a section with the default specifications. Note, that the relevant subsections will not be created recursively.
  - **Copy file into a section** This option allows you to copy data structures from external files. It works like this: the model will copy the data from an external file and paste it in your input model file.
  - **Include file as a section** This option also allows the usage of external data files. But it works slightly different: the model will only use the external data without writing the whole data it in your model file. Instead, you will see in your input file in the relevant section a line like this: Include filename.in SectionName. This allows to have only one definition of an argon mixture, say, on disk, which can be shared (included) by all models that describe some argon plasma.

### 3.3.2 The Leaf Editors

At the right-hand side of the model editor window, an appropriate editor for the data items of the active node are shown. Depending on the node contents, you will see a **Dialog editor**, a **Grid editor** or a **Text editor** window. In figure 2.2 a dialog editor is visible. The various editors are displayed in figure 3.1 and will be discussed next.
Figure 3.1: The various leaf editor windows. Figure 3.1(a) shows the dialog editor. The example shows the time settings of a time-dependent model edited using the dialog editor. Figure 3.1(b) shows a text editor, as an example is a lookup table. Figure 3.1(c) shows the grid editor in action (here a PDP geometry is being edited).

The Dialog Editor

The node dialog editor window (figure 3.1(a)) is the one you will most commonly encounter. It contains one leaf block for each data item type. The name of the data item and a short description are displayed above the box. Note that a box may be empty. Clicking on the name of the data item with the right mouse button pops up a menu which allows you to add a new leaf of the relevant type. The default values will be used.

Each entry has a check box on the left, followed by one or more editor or choice fields. By clicking
on the check box with the left mouse button, the entry can be disabled or enabled. Disabled entries are
not taken into account when the model description is interpreted by PLASIMO. They appear in grey
and cannot be edited.

In addition, two small text editor windows are available. The ‘Annotation’ widget can be used to
provide a description of the active section. If one exists, it will be shown. The window titled 'Other
entries' can be used to add entries in text raw form. PLASIMO also dumps entries it cannot handle here
when creating a dialog editor, so it should be empty after creating a fresh dialog editor.

The Grid Editor

The grid editor (figure [3.1(c)]) facilitates editing of grids (“matrices”) consisting of elements which can
be represented as strings. Sometimes the matrix is too big to be visualized in the grid editor. You can
zoom out/zoom in the matrix by pressing the control key and scrolling the mouse wheel. The matrix
consists on cells with different colors as to each color is assigned different material type.

There are two editing modes, the Grid layout mode and Grid cell mode. By default, the cell
mode is active. One can switch between these modes via the context menu which is activated by pressing
the right mouse button on the grid region.

The Grid cell mode deals with the properties of the cells, such as assignment of new material type.
Clicking or selecting cells with the left mouse button will assign the active color to them (by default this
is the color black, which always corresponds to the material with the lowest alphabetic key value). If the
control key is down, instead the material under the left mouse button is picked up as active color. New
material color-index pairs can be created via the menu which is activated by clicking the right mouse
button and selecting Select material. A list of previously defined materials will pops up for selection
of material type.

In Grid layout mode only entire rows or columns can be selected. Row and column selection can be
activated by clicking on the grey label bar at the left and top side of the grid, respectively. The presently
selected row(s) or column(s) can be cut and copied via the menu which is activated by clicking the right
mouse button. Then, the cut or copied row(s)/column(s) can be pasted before the active row/column.

Finally, after editing the geometry matrix, you may want to keep or discard the changes. The changes
will be automatically saved after switching to another node. If you do not want to save the changes you
made, you can retrieve the original matrix by pressing the right mouse button on the CellMatrix in
the tree editor and selecting Undo all changes from the menu. Note, that after switching to another
node, editing can no longer be undone.

The Text Editor

The text editor window allows the user to manipulate the data entries in raw format. In some cases
the creation of a text editor window is requested explicitly, because the nature of the data does not
allow convenient editing inside a node dialog or grid editor dialog. Perhaps an appropriate editor will be
provided later. An example is a look up table in a two-column format (see figure [3.1(b)]).

The text editor is also used as fallback option if the creation of a node dialog or grid editor window
fails for some reason. If the latter happens, the user will be informed and is kindly requested to send a
bug report to the PLASIMO maintainers.

There are a few things to think of when using the text editor window. Firstly, disabled leafs occur in
raw text format with an underscore (_) prepended. Secondly, make sure to quote items inside this entry
which consist of multiple words. As an example, an entry which specifies a name should be written as
Name "Jan van Dijk" instead of Name Jan van Dijk. In the latter case, you will perhaps get an error
when PLASIMO interprets this entry, but more likely the part van Dijk will be silently ignored. The
text control correctly handles special characters, so newlines, tabs and backslashes can be safely used.
3.4 The Data Window

After installing the model, an additional tab window has appeared, named Model data. You will see that the available data in this window have been organized as a tree, displayed on the left. The tree consists of a model-dependent data, which may be grouped into sections of related variables. As a minimum, a section called General has been created for every PLASIMO model. Not surprisingly, its contents provide some general information about the model, such as the status of the calculation (you may think of data like elapsed time and the number of iterations). This can be activated by double clicking on the item common data inside section General.

![Figure 3.2: A screen-shot of PLASIMO’s graphical user interface (GUI). The screen-shot has been made during a run of the model cylindrical_2d_atmospheric_pressure.gum and shows an OpenGL plot of the temperature of the plasma region.](image)

For each data item a “viewer” can be created. Creating a viewer can be easily achieved by moving the mouse over the data item ‘temperature’, say, and double click on it will open the default OpenGL viewer and you get the result shown in figure 3.2. Please take a moment to see what data are there.

By default, all viewers are embedded at the right hand side of the data window, and are displayed as a series of tab windows. If you would like to monitor the viewers simultaneously, just drag the tab to the bottom or to the right of the data window.

PLASIMO offers the option to open all the viewers in separate windows (not embedded in the application). You can achieve by selecting option EmbeddedViewers no in the GlobalSetting menu before installing a model. If you would like to keep this behaviour permanently, you can change the setting EmbeddedViewers in the configuration file runtimecnf.in. The file is located in folder plasimo_root_directory/config/.

Additional viewer types are available and you can choose one by clicking the right mouse button in the selected item. This will pop up a menu which offers the option ‘Add view’. Some additional
functionalities of the various viewers are discussed in the following section.

3.4.1 Additional Functionalities of the Viewers

Some additional functionalities of the various viewers are available. The next text discusses the different viewers available in PLASIMO. Depending on the data item different viewers are available. PLASIMO will offer you the appropriate suggestions for the relevant data item.

**OpenGL**

The OpenGL viewer is the default viewer in PLASIMO for plotting 2D and 3D plots. The plot can be rotated by pressing the left mouse button on the plot and drag the plot until the final orientation is reached. The same effect can be obtained using the arrow keys. If you keep the shift key pressed during such operations, the plot will be translated, rather than rotated. The plot window can also be controlled with the following keys:

- f – Switch between solid and wire frame plot.
- r – Reset the configuration parameters to their default values.
- L – Toggle lighting.
- s – Toggle smoothing.
- o – Include the origin in the plot.
- p – Print the plot window to a postscript file.
- d – switch between the directions
- c – switch between the components
- C – Toggle enabling (all) clipping planes
- <, > – Decrease/increase the number of planes that is drawn (3D only)
- X,Y,Z – This is available only for 3D plots. Draw planes with constant first, second or third coordinate.
- x, y, z – Activate the x, y or z coordinate. All subsequent coordinate-specific commands will affect that coordinate. The commands below are all coordinate-specific:
  - a – Increase the position of the lower clipping plane.
  - A – Decrease the position of the lower clipping plane.
  - b – Increase the position of the upper clipping plane.
  - B – Decrease the position of the upper clipping plane.
  - l – Toggle between linear and logarithmic axes.

**MathPlot WireFrame**

This viewer is suitable for plotting data in a X-Y format. This is the default viewer in PLASIMO for plotting this type of data. Pressing the right mouse button, the pop up menu will give you a list of all options available for this plot.

**Text view**

The viewer shows the data in text format.

**Grid**

This viewer shows the data for one field variable as a spreadsheet. The cells in the spreadsheet are placed to resemble the positions of the corresponding data field variable in the master grid. This means that for a cell showing the value of a field defined at the nodal points (e.g. the species densities), the north, south, west and east cells are empty, because in the master grid they represent the positions of the flux
field variables. See the grid document for more information.

3.4.2 Initial Viewers

Sometimes, in the working process you may need to install a model multiple times and to monitor some data item at different conditions for example. After each Install of the model you have to open the relevant viewer to monitor the data item in run time. This could be annoying sometimes, so that PLASIMO offers the option to avoid this by setting initially the viewers of the variables you want to monitor. As a result, when you install the model, the viewers will be initially open with the set of specifications.

Assume that you want a viewer for “temperature” data item, as it is shown in figure 3.2 to be initially open always when you install the model. This can be done by adding the optional section InitViews in the model tree. Press with right mouse button on the Model and select Add InitViews from the pop up menu. Next, you have to add the relevant subsection: press the right mouse button on InitViews and add default section ViewList. On the leaf dialog editor on the right hand side of the window press again the right mouse button on the Viewer leaf to add one. Adding a new viewer leaf will displays 4 empty boxes. In the first one you have to specify the data location, for example vessel/interior/Temperature. In the second box you have to specify the name of the data item temperature (K). In the third you can specify the viewer type. If it is empty, the default viewer will be open. In the last box optionally you can specify some attributes, such as minimum and/or maximum values of the x, y or z coordinate. All the attributes must be separated by comma, for instance:

\[
\begin{align*}
x_{\text{min}} &= 0, \quad x_{\text{max}} = 0.05, \quad y_{\text{min}} = 0, \quad y_{\text{max}} = 0.005, \quad z_{\text{min}} = 1.5e3, \quad z_{\text{max}} = 4e3
\end{align*}
\]

For x-y data plots the attribute can be logarithmic scale. For instance, if you would like to visualize the convergence log with logarithmic y axes, the leaf Viewer should look like this:

You can add viewers leafs as many as you can/want. Note that the editor is case sensitive!

3.5 Model-specific GUI Extensions

In section 2.3 it was mentioned that in addition to the functionality which is shared by all GUM simulation types, a number of application-specific extensions may be present. There are two such extensions available:

- Grid resizing: the grid can be re-dimensionalized on the fly;
- Adjustment of relaxation factors: PLASIMO models allow the inspection and adjustment of the iterative control parameters while a simulation is in progress.

If the model supports these functionalities, after installing it, you will see an additional menu called Controls. This menu provides access to the extensions. If the extensions are not relevant for your model, the menu item is greyed out. As an example, the relaxation factor control is disabled if there are no relaxation factors in your model.

Grid Resizing

Grid resizing can be seen as a poor man’s implementation of the multi-grid algorithm. Essentially, in such methods the number of grid cells is not kept constant. Rather, the results of rapidly converging but inaccurate calculations on coarse meshes are combined with accurate, yet slowly converging calculations
on finer meshes. The result is a net reduction of the time-to-convergence. Typically a number of grid changes is made in both directions. A full multi-grid algorithm has not been implemented in PLASIMO. However, wxPlasimo allows its users to manually adjust the mesh size while a simulation is in progress. After a grid resize all grid-dependent variables are interpolated to the new grid and the iterative process resumes.

Please note that this grid resizing can also be configured to be done automatically at the input file level, see section MainIterConfig of chapter ‘Input File Reference’ of the User Reference Guide for details.

Relaxation Factor Adjustment

This allows the adjustment of the iterative control variables of which are part of the model. More specifically, the user is allowed to set the actual value of the under-relaxation factor (URF), of the automatic relaxation factor adjustment rate, and of the maximum URF value that will be configured by this automatic process. The URF is the fractional amount of a correction that is actually applied to the old value of a variable. Hence, a URF of unity means that the correction will be completely applied, while for smaller values only a partial adjustment is done. There is a trade-off between speed and robustness here: while smaller under-relaxation factors tend to stabilize the simulation, the time-to-convergence may be considerably longer.

If you select “Relaxation factor adjustment” from the Controls menu once a model is installed, a dialog appears that allows you to select a relaxation factor object. If one is selected, a second box appears that allows to change the value of the under-relaxation parameters; the actual value is proposed as default. Note that most variables rely on a non-zero URF value.
Chapter 4

Custom expressions in PLASIMO

4.1 Custom expressions

PLASIMO allows user-defined expressions and it can perform mathematical operations on a number of variables and constants, which can be declared, and then make the results available like it does for standard variables. It is also possible to use a custom-defined field for the computation of another.

The syntax of the user-defined expressions is inspired by the C/C++ expression grammar. It provides support for almost all operators that can also occur in C/C++ expressions and follows the same operator precedence rules. The supported operators and their precedence rules are listed in table 4.1. For those who are not familiar with C/C++ expressions, we point out that the ternary operator a ? b : c is defined such that it evaluates to b if a is non-zero (‘true’) and to c otherwise.

4.2 Supported Operators and Functions

Like in C and C++, not all operations or functions can be used with all value types. For example, for complex data types, comparison operators are available, but compare the real parts of the values. An error is generated when the imaginary parts are non-zero. This leaves it up to the user to decide what he means when he tries to compare complex numbers and allows him to write something like abs(z1)<abs(z2) or real(z1)<real(z2), as he sees fit.

For the unsigned, int, double and complex double data types the usual collection of functions is available, like the trigonometric functions sin, exponential and logarithmic functions. Table 4.3 presents an overview of the available functions.

4.2.1 Variables that can be used in expressions
<table>
<thead>
<tr>
<th>symbol</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>()</td>
<td>grouping, function calls</td>
</tr>
<tr>
<td>[]</td>
<td>array indexing</td>
</tr>
<tr>
<td>x^y</td>
<td>exponentiation, ( x^y )</td>
</tr>
<tr>
<td>+x</td>
<td>unary plus: ( +x ) is equal to ( x )</td>
</tr>
<tr>
<td>-x</td>
<td>negation</td>
</tr>
<tr>
<td>~x</td>
<td>bitwise not</td>
</tr>
<tr>
<td>x * y</td>
<td>multiplication</td>
</tr>
<tr>
<td>x / y</td>
<td>division</td>
</tr>
<tr>
<td>x % y</td>
<td>remainder</td>
</tr>
<tr>
<td>x + y</td>
<td>addition</td>
</tr>
<tr>
<td>x - y</td>
<td>subtraction</td>
</tr>
<tr>
<td>x &lt;&lt; y</td>
<td>bitwise left shift</td>
</tr>
<tr>
<td>x &gt;&gt; y</td>
<td>bitwise right shift</td>
</tr>
<tr>
<td>&lt;</td>
<td>comparison</td>
</tr>
<tr>
<td>&lt;=</td>
<td>comparison</td>
</tr>
<tr>
<td>&lt;</td>
<td>comparison</td>
</tr>
<tr>
<td>&gt;=</td>
<td>comparison</td>
</tr>
<tr>
<td>x &amp; y</td>
<td>bitwise and</td>
</tr>
<tr>
<td>x !</td>
<td>y</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>a &amp;&amp; b</td>
<td>logical-and</td>
</tr>
<tr>
<td>a</td>
<td></td>
</tr>
<tr>
<td>a ? b : c</td>
<td>if-then-else</td>
</tr>
<tr>
<td>x = y</td>
<td>declares ( x ), assigns ( y )</td>
</tr>
<tr>
<td>x , y</td>
<td>equals ( y ), but first ( x ) is evaluated</td>
</tr>
</tbody>
</table>

Table 4.1: Operators, from high to low precedence. Operators within the same block have the same precedence. Note that the caret is used for exponentiation, whereas C(++) users would expect it to denote an exclusive-or. Also the assignment operator is different, since it declares a new identifier \( x \).
## Constants

<table>
<thead>
<tr>
<th>symbol</th>
<th>comment</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\pi</code></td>
<td>π</td>
<td>3.1415926535897932384626433833</td>
</tr>
<tr>
<td><code>\c</code></td>
<td>speed of light</td>
<td>2.99792458 × 10^8 m/s</td>
</tr>
<tr>
<td><code>\G</code></td>
<td>gravity constant</td>
<td>6.67408 × 10^{-11} kg^{-1} m^3 s^{-2}</td>
</tr>
<tr>
<td><code>\me</code></td>
<td>electron mass</td>
<td>9.10938356 × 10^{-31} kg</td>
</tr>
<tr>
<td><code>\mp</code></td>
<td>proton mass</td>
<td>1.672612898 × 10^{-27} kg</td>
</tr>
<tr>
<td><code>\qe</code></td>
<td>elementary charge</td>
<td>1.6021766208 × 10^{-19} C</td>
</tr>
<tr>
<td><code>\kB</code></td>
<td>Boltzmann constant</td>
<td>1.38064852 × 10^{-23} kg m^2 s^{-2} K^{-1}</td>
</tr>
<tr>
<td><code>\h</code></td>
<td>Planck’s constant</td>
<td>6.022140857 × 10^{23} mol^{-1}</td>
</tr>
<tr>
<td><code>\hbar</code></td>
<td><code>\h/(2\pi)</code></td>
<td>1.05457 × 10^{-34} kg m^2 s^{-1}</td>
</tr>
</tbody>
</table>

| `\hbar` | `\h/(2\pi)` | 1.05457 × 10^{-34} kg m^2 s^{-1} |

### SI base units

<table>
<thead>
<tr>
<th>base</th>
<th>alternative</th>
</tr>
</thead>
<tbody>
<tr>
<td>kg</td>
<td>g, amu</td>
</tr>
<tr>
<td>m</td>
<td>Ang, in</td>
</tr>
<tr>
<td>s</td>
<td>minute, hour, day, year</td>
</tr>
<tr>
<td>Å</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>eV T (=11604.5 K)</td>
</tr>
<tr>
<td>mol</td>
<td></td>
</tr>
<tr>
<td>cd</td>
<td></td>
</tr>
<tr>
<td>rad</td>
<td>deg (=0.0174533 rad)</td>
</tr>
<tr>
<td>sr</td>
<td></td>
</tr>
</tbody>
</table>

### Derived units

<table>
<thead>
<tr>
<th>symbol</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1 s A</td>
</tr>
<tr>
<td>F</td>
<td>1 kg^{-1} m^{-2} s^4 A^2</td>
</tr>
<tr>
<td>H</td>
<td>1 kg m^2 s^{-2} A^{-2}</td>
</tr>
<tr>
<td>Hartree</td>
<td>4.35974 × 10^{-18} kg m^2 s^{-2}</td>
</tr>
<tr>
<td>Hz</td>
<td>1 s^{-1}</td>
</tr>
<tr>
<td>J</td>
<td>1 kg m^2 s^{-2}</td>
</tr>
<tr>
<td>N</td>
<td>1 kg m s^{-2}</td>
</tr>
<tr>
<td>Ohm</td>
<td>1 kg m^2 s^{-3} A^{-2}</td>
</tr>
<tr>
<td>Pa</td>
<td>1 kg m^{-1} s^{-2}</td>
</tr>
<tr>
<td>S</td>
<td>1 kg^{-1} m^{-2} s^3 A^2</td>
</tr>
<tr>
<td>T</td>
<td>1 kg s^{-2} A^{-1}</td>
</tr>
<tr>
<td>Td</td>
<td>1 × 10^{-21} kg m^4 s^{-3} A^{-1}</td>
</tr>
<tr>
<td>Torr</td>
<td>133.322 kg m^{-1} s^{-2}</td>
</tr>
<tr>
<td>V</td>
<td>1 kg m^2 s^{-3} A^{-1}</td>
</tr>
<tr>
<td>W</td>
<td>1 kg m^2 s^{-3}</td>
</tr>
<tr>
<td>Wb</td>
<td>1 kg m^2 s^{-2} A^{-1}</td>
</tr>
<tr>
<td>atm</td>
<td>101325 kg m^{-1} s^{-2}</td>
</tr>
<tr>
<td>bar</td>
<td>100000 kg m^{-1} s^{-2}</td>
</tr>
<tr>
<td>cmH</td>
<td>1.98645 × 10^{-23} kg m^2 s^{-2}</td>
</tr>
<tr>
<td>eV</td>
<td>1.60218 × 10^{-19} kg m^2 s^{-2}</td>
</tr>
<tr>
<td>l</td>
<td>0.001 m^3</td>
</tr>
<tr>
<td>lm</td>
<td>1 cd sr</td>
</tr>
<tr>
<td>lx</td>
<td>1 m^{-2} cd sr</td>
</tr>
<tr>
<td>sccm</td>
<td>4.47797 × 10^{17} s^{-1}</td>
</tr>
<tr>
<td>sccs</td>
<td>2.68678 × 10^{19} s^{-1}</td>
</tr>
</tbody>
</table>

Table 4.2: Constants and units available in PLASIMO.
<table>
<thead>
<tr>
<th>symbol</th>
<th>comments</th>
<th>uint</th>
<th>int</th>
<th>double</th>
<th>std::complex&lt;double&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs(x)</td>
<td>$</td>
<td>x</td>
<td>$, absolute value of $x$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>min(x,y)</td>
<td>smallest of $x$ and $y$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓†</td>
</tr>
<tr>
<td>max(x,y)</td>
<td>greatest of $x$ and $y$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓†</td>
</tr>
<tr>
<td>acos(x)</td>
<td>arc cosine of $x$</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>asin(x)</td>
<td>arc sine of $x$</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>atan(x)</td>
<td>arc tangent of $x$</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>floor(x)</td>
<td>largest integer not greater than $x$</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ceil(x)</td>
<td>smallest integer not less than $x$</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fmod(x,y)</td>
<td>remainder of $x/y$</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cos(x)</td>
<td>cosine of $x$</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>cosh(x)</td>
<td>hyperbolic cosine of $x$</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>exp(x)</td>
<td>$e^x$</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>imaginary unit</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>ln(x)</td>
<td>natural logarithm of $x$</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>log(x)</td>
<td>equal to $\ln(x)$</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>log10(x)</td>
<td>base-10 logarithm of $x$</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>pi</code></td>
<td>mathematical constant $\pi$</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sin(x)</td>
<td>sine of $x$</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sinh(x)</td>
<td>hyperbolic sine of $x$</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sqrt(x)</td>
<td>$\sqrt{x}$, square root of $x$</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tan(x)</td>
<td>tangent of $x$</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tanh(x)</td>
<td>hyperbolic tangent of $x$</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>real(z)</td>
<td>real part of complex value $z$</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>imag(z)</td>
<td>imaginary part of complex value $z$</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>arg(z)</td>
<td>phase angle of complex value $z$</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.3: Functions that are available for `int`, `double` and `std::complex<double>` value types. †: these functions require that the imaginary parts of both arguments are zero and operate on the real parts if that is the case.
<table>
<thead>
<tr>
<th>symbol</th>
<th>GM</th>
<th>LTE</th>
<th>2T</th>
<th>DDI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Species</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Density('&lt;species&gt;')</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MassFraction('&lt;species&gt;')</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Mass('&lt;species&gt;')</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Charge('&lt;species&gt;')</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Energy('&lt;species&gt;')</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TotalDensity()</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MassDensity()</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Species transport</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diffusion('&lt;species&gt;')</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mobility('&lt;species&gt;')</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FluxDensity('&lt;species&gt;')</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElectronMobilityN()</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElectronDN()</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Flow</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pressure()</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LaminarViscosity()</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MassDens()</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>EM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElectricalConductivity()</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RelativePermittivity()</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OhmicDissipation()</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Potential()</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>ElectricField()</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>SurfaceChargeDensity()</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VolumeChargeDensity()</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Temperature</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temperature()</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temperature('&lt;species&gt;')</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>GasTemperature()</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ElectronTemperature()</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElasticEnergyTransfer()</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ThermalConductivity()</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ThermalConductivityHP()</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ThermalConductivityEL()</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4: Variables that are available and can be used in custom expressions in the Global model (GM), LTE, quasi-neutral 2-temperature model (2T) and the drift-diffusion-inertia (DDI) model. Note that the variables are available for the various models only if they are calculated.
Chapter 5

PLASIMO Example Models

5.1 Introduction
This chapter explains some of the existing demo models that were shipped with the PLASIMO distribution. The examples give also information how you can edit an existing model and create a new one using the graphical user interface.

5.2 Drift-Diffusion Models

5.2.1 Introduction
The PLASIMO's drift-diffusion module is a time-dependent multi-fluid model that is based on balance equations derived from the Boltzmann equations. One of the main assumptions in the drift-diffusion model is that the background gas is dominant. For the defined active species (electrons, ions and (excited) atoms, molecules) we solve the balance equations.

The temporal and spatial evolution of the density $n_p$ of each active species $p$ is described by the particle balance:

$$\frac{\partial n_p}{\partial t} + \nabla \cdot \vec{\Gamma}_p = S_p, \quad (5.1)$$

where $S_p$ is the net source term determined by the reactions occurring in the discharge. The flux density $\vec{\Gamma}_p$ describes the transport of particles due to drift and diffusion:

$$\vec{\Gamma}_p = \mu_p \vec{E} n_p - D_p \nabla n_p \quad (5.2)$$

where $\mu_p$ is the mobility of species $p$, $\vec{E}$ the electric field and $D_p$ the diffusion coefficient. The mobility is taken to be negative for negatively charged species and zero for neutral particles.

The electron energy balance is determined by:

$$\frac{\partial n_\varepsilon}{\partial t} + \nabla \cdot \vec{\Gamma}_\varepsilon = S_\varepsilon \quad (5.3)$$

with electron energy flux density given by:

$$\vec{\Gamma}_\varepsilon = \frac{5}{3} \mu_e \vec{E} n_\varepsilon - \frac{5}{3} D_e \nabla n_\varepsilon. \quad (5.4)$$

Here $n_\varepsilon = n_e \varepsilon$ is the electron energy density and $\varepsilon$ is the mean electron energy.
The effective source term $S_e$ for electron energy is given by:

$$S_e = \vec{\Gamma}_e \cdot \vec{E} - n_e \sum_r \varepsilon_r k_r n_r - L_e,$$

where the first term represents heating by the electric field and the second the energy loss in inelastic collisions. The summation runs only over the electron impact reactions, with $n_e$ the density of the target particles and $\varepsilon_r$ the threshold energy. The last term $L_e$ in the equation represents the energy loss due to elastic collisions.

These equations are coupled with the Poisson equation for computation of the electric field. The transport equations — the continuity equation (5.1), the momentum balance equation (5.2), and the electron energy balance equation (5.3) — are coupled to Poisson equation for computation of the electric field:

$$\nabla \cdot (\epsilon \vec{E}) = -\nabla \cdot (\epsilon \nabla V) = \rho,$$

where $\epsilon$ is the dielectric permittivity, $V$ the electrostatic potential, and $\rho$ the space charge density given by:

$$\rho = \sum_p q_p n_p.$$

Poisson equation is solved in the plasma region and the dielectric regions, in order to account for the influence of charged species on the electric field. Note that $\rho$ may be discontinuous and that surface charges may be present.

More information about the governing equations and their derivation can be found in [2, 3].

Boundary conditions

The default boundary conditions in the drift-diffusion module are: At the symmetry axis and the open boundaries homogeneous Neumann BC are employed, meaning that the derivatives of the quantities in the directions perpendicular to these boundaries are set to zero. For the potential distribution at the electrodes, the expressions of the applied voltages are used (Dirichlet boundary conditions). The BC for $n_p$ and $n_e$ at the physical boundary are given by expressions for the flux densities $\Gamma_p$ and $\Gamma_e$:

$$\vec{\Gamma}_p \cdot \vec{n} = (1 - r)n_p \left( a_p \mu_p E_p \cdot \vec{n} + \frac{1}{4} v_{th} \right) - S_p$$

where $r$ is the reflection coefficient, $v_{th} = \sqrt{8kT/\pi m}$ the thermal velocity. The parameter $a_p$ is set to 1 if the drift velocity is directed to the wall and zero otherwise:

$$a_p = \begin{cases} 1, & \mu_p E_p \cdot \vec{n} > 0 \\ 0, & \mu_p E_p \cdot \vec{n} < 0 \end{cases}$$

The source term $S_p$ account for wall production/destruction at the wall, such as secondary emission, sputtering, desorption, thermo-ionic emission. These wall processes are user-specified in section WallProcessList.

Optionally the above-described boundary conditions can be overwritten by adding an optional section BoundaryConditions.

Module options

Optionally the drift-diffusion model described above can be consistently coupled with gas temperature calculations. We solve the energy balance equation in terms of heavy particle temperature.

Optionally, the effect of the gas flow is described with solving the mass balance equation for the heavy particles together with the Navier-Stokes equation (bulk momentum balance). The time rate of
change of the mean momentum, in each fluid element, is due to the net pressure forces, viscous stress, the Lorentz force and gravity. The pressure contribution is due to gradient in the pressure, rather than the pressure itself. The absolute pressure is dealt with the ideal gas law. For the viscous part of the stress tensor we apply Stokes model: This is applicable for laminar flows. For turbulent flows, there is an additional turbulent stress term which requires a turbulence model. In PLASIMO $\kappa-\epsilon$ and $\kappa-\omega$ models are available.

This section presents example input files and shows how the drift-diffusion module is used.

5.2.2 DC Glow Discharge Tube

Example input file: demo_fl.gum
Location: plmd2d_v2/input/

Model specifications
This input file contains a model of a low pressure DC discharge with a simple geometry: two parallel electrodes at a distance of 45 cm, enclosed in a glass tube with a inner radius of 3.2 cm. The gas is He and only electron impact ionization is considered. The gas pressure is constant at 1 Torr, and a constant voltage of $-400$ V is applied.

Install and run the model
After installing the model, an additional tab “Model data” appears in the main window. On the left hand side in the herarchial tree you will find all the variables that are calculated in the model. The output data is organized in sections and subsections where the related variables rely. A plot of a variable can be activated by double-clicking. The following sections are present:

- **Discharge Region:**
  This section contains all variables that are calculated inside the plasma region. The section is subdivided into:
  - **EM Data:**
    In this section you can see the EM data that is calculated on the discharge region. That is (reduced) electric field, power dissipation and current density.
  - **Electron Energy:** contains the data from the energy equation, such as the mean electron energy, mean energy source, mean energy flux density.
  - **Gas Temperature:** contains the gas temperature related variables.
  - **Mixture:** contains Species (variables for each species, such as density, mobility, source, flux) and Reactions (rates of the specified reactions).
  - **History Data:** This section is available if it is requested from the input. It contains plots of user-specified variables as a function of time, averaged over the volume. In this example, the variables are the species densities.

- **EM Region:**
  In this section you can see the EM data, such as potential distribution, surface and volume charge densities.

- **General:**
  Naturally contains the general common data, such as the status of the simulation. You can see the model status by double-clicking on common data.

- **Geometry:**
  Here you can see the geometry - the configuration and the control volume grids.
Results

When the model is loaded, installed, and run, it will be possible to identify several regions in the discharge:

- **Cathode fall region (Cathode dark space):**
  Most of the potential drop between the electrodes occurs in the cathode dark space. This is a region with a strong electric field due to the positive space charge in front of the cathode. The strong field accelerates ions toward the cathode and electrons toward the negative glow region. The electrons ejected from the cathode are accelerated and gain energies up to the cathode fall.

  Select the **EM Region** node to visualize the potential, the volume and surface charge densities; the **Discharge Region -> Mixture -> Species** to monitor the flux density of the ions and electrons; and **Electron Energy** to visualize the mean electron energy.

- **Negative glow:**
  In the negative glow an intense electron impact ionization occurs. The electrons lose their energy and the E-field drops to almost zero. Select the **Discharge Region -> Mixture -> Reactions** node for the reaction rate and **Discharge Region -> Mixture -> Species -> e** to see the power dissipation of the electrons.

- **Faraday dark space:**
  As the electrons have dissipated their energy, excitation and ionization will become less and less frequent, because electrons do not gain energy in the weak field. In this region the longitudinal field gradually increases to the E-field in the positive column.

- **Positive column:**
  In this region the potential gradient is practically constant, and electron impact ionization occurs throughout this region.

Output files

All calculated variables are not only shown in the GUI but are also stored on disk. The data path as well as the number of writes of the output data can be specified in the input (see MainIterConfig). By default, the program store the output files in the main PLASIMO folder if no data path is specified in the leaf [MainIterConfig]->Directory. In subsection [MainIterConfig]->[TimeSchedule]->[ScheduleBlock] the leaf dt_log specifies the period of writing the output data.

The spatial distribution of each quantity is written in a separate text file. The files names are constructed as follows:

\[
<\text{region name}>_<\text{variable name}>_<\text{suffix}>.<\text{extension}>
\]

In this example two calculation regions are available: the plasma region that contains only material with ID 0, and interior region that contains materials with ID 0 and 2 (dielectric). The variables that are calculated on the plasma regions starts with **plasma_**, while the variables that are calculated on the interior region starts with **interior_**.

For some of the calculated variables the data is printed per species. These variables are the density, diffusion, mobility, flux density, source and power dissipation. The \(<\text{variable name}>\) then consists of he variable’s name followed by the index of the species. The indexing of the species starts from 0 and matches the order of the species specified in the input. In this case **density_0** contains the density distribution of the electrons while **density_1** is the density distribution of the second defined particle, in this case He\(^+\).

The **<Prefix>** is a user-specified prefix (leaf [MainIterConfig]->Prefix). Changing the prefix can help you to distinguish output from different calculations. In this example, the prefix is case1. Thus, the full name of the output file that contains the electron density data is **case1_plasma_density_0.dat**.
Apart from the spatial distribution of the variables, a number of additional output files are produced:

- **history.out**: volume averaged quantities as a function of time. The quantities are written with output refresh rate as specified in `LogInterval` in section `MainIterConfig`.

- **status.dat**: the status of the calculation together with some variables averaged over the volume; the file is overwritten at every time step.

- **status_history.out**: the same data as in `status.dat`, but written at a user-specified period.

- **reaction_analysis.out** and **reaction_analysis.txt**: detailed reaction analysis: production and destruction contribution of each reaction for each species. The files are written like `status.dat` and `status_history.dat`, respectively.

### Additional output files on user's request

Various output data can be configured in the input.

- Volume averaged time-dependent data
  This data can be requested by adding an optional section `HistoryData` where you can specify which data should be recorded as a function of time.

### 5.2.3 Sputtering Hollow Cathode Discharge

**Example input file:** `hcd-demo.gum`  
**Location:** `plmd2d_v2/input/`

#### Model specifications

The input file contains a model of a hollow cathode discharge (HCD) used for laser application. Description of the complete model you can find in [5, 6, 2], while here is a simplified version for demonstration purposes.

The geometry is a cylindrical hollow cathode with 4 mm inner diameter and 50 mm length; two anode rings are placed at both sides of the cathode, separated from each other with quartz rings. The geometry is symmetric around the center of the cathode (both axially and azimuthally). Hence, the simulated discharge region is half of the cathode length, the anode region and the dielectric in between. A fixed voltage of -300 V is applied.

The operating gas is He at constant pressure of 2.3 kPa. The considered species are He\(^+\), Cu and Cu\(^+\). The buffer gas atoms are ionized by electron impact ionization. The metal atoms in the discharge are produced only by sputtering due to ion bombardment of the cathode surface. The metal atoms are ionized in the plasma and they also can produce sputtering (self-sputtering). The ionization mechanism of Cu includes electron impact ionization and charge transfer with buffer gas ions.

![Figure 5.1: The hollow cathode geometry.](image_url)
Sputtering module

The sputtering process can be taken into account in the model and it can be applied at different materials if multiple materials are present. This can be done as follows:

- In section PlasmaRegion with the right mouse bottom add section WallProcessList
- In WallProcessList you must add the process WallProcess of type sputtering.
- Select WallProcess; on the right side window you can fill the species bombarding the wall (the projectile) and the sputtered atoms (recoils). You have to specify also the material where the process should be applied. The names of the materials are defined in section Geometry. The sputtering process is applied on materials with names cathode and anode.
- The sputtering yield for every projectile species is specified by adding a subsection SputteringYield. The SputteringYield is of type Constant.

5.2.4 Building a 1D Drift-Diffusion Model from Scratch

Model specifications

This model simulates an argon glow discharge created between two infinitely large parallel planar electrodes spaced 2 cm apart. Therefore, it is a one dimensional model. Over the electrodes a RF voltage is applied with an amplitude of 40 V and a frequency of 13.56 MHz. The background gas argon is assumed to be weakly ionized and its temperature and density remain unchanged. The model input parameters are taken from previously published works and the results can be directly compared. Table 5.1 summarizes the input data for this model taken from ref [7]. The ionization rate coefficient is based on the experimental data presented in [7] it reads:

$$K_{ion} = \begin{cases} 
8.7 \times 10^{-15} (\varepsilon/eV - 5.3) \exp \left( -4.9/\sqrt{\varepsilon/eV - 5.3} \right) & \varepsilon > 5.3 eV \\
0 & \varepsilon \leq 5.3 eV 
\end{cases} \quad (5.10)$$

where $\varepsilon$ is the mean electron energy. In this model only electron impact ionization will take place, other elastic and inelastic processes are not taken into account.

The boundary conditions for this model are as follows: For the ion densities Homogeneous Neumann BC are applied on both electrodes, for the electron density and electron energy density homogeneous Dirichlet BC are applied, and finally for the potential boundary conditions are the applied voltages
(Dirichlet BC). In PLASIMO’s drift diffusion model flux boundary conditions are used by default. We will see later how the default boundary conditions can be overwritten in order to achieve the desired model behavior.

**Model construction**

We will begin by creating the computational grid, followed by the species and their reactions, and finally the model control parameters.

Whenever the text says to create a new section and the section contains some text between square brackets, for instance `NewSection[SomeType]` the text between the brackets is the Type of the section. This means that in the submenu of the right-click menu the appropriate entry must be selected. At all times the help window can give you more information about the section of the model currently selected.

1. Start PLASIMO as described in section 2.2.
2. Click Create new model (left icon, or use the menu) and select DriftDiffusion-1D.
3. In Model add !1D!Grid[cartesian].
4. Set $x_{\text{max}}$ to 0.02*m.
5. Add section Geometry and set the entry for BaseRefinementPower to 5. The help window gives you more information about the grid refinement.
6. In Geometry add 3 Material sections named plasma, grounded and powered. Set the Id in each of the materials starting with 0 for the plasma, 1 for the anode and 2 for the cathode.
7. In Material plasma add a property by right-clicking in the Property frame, with key Type and value DischargeGas.
8. In both the grounded and powered sections add two Property key-value pairs, namely Type — Electrode.
9. Add section CellVector and set the materials in the matrix (see the user guide on how to use the grid material editor). By default you will see a cell vector containing the Id’s of the defined materials: 102
10. Add a section Stretch[TwoPt_Stretch]. That means that a stretching of the numerical grid will be applied around two points, logically more grid points in the sheath regions, i.e. close to the electrodes. On the leaf editor, specify for Point1 0.0 and Density1 2; and for the second point Point2 1.0 and Density2 2.
11. In Model add PlasmaRegion[DriftDiffusion]. The section is created with one subsection which is not used for this model.
12. Set RegionId to interior.
13. Set the pressure to 1*Torr. The right-hand side pull-down menu can be used to select more convenient units.
14. In PlasmaRegion add GasTemperature[Fixed]. In the default created subsection TemperatureFunction the value is set by default to 300K.
15. Add section Mixture[DriftDiffusion]. The section is created with three required subsections: GasList, SpeciesList and ReactionList[List].
16. In section GasList add Gas and set Name to Ar.
17. In Gas add State[Atom] and set Name to ground.
18. In SpeciesList add subsection Species. Set name to $\text{Ar}^+$, InitDens to $5 \times 10^{15} \text{m}^{-3}$.

19. In the Species add a State[Atom] named ion with Energy to $15.578 \text{eV}$, and Weight to 6.

20. In the Species we have to add the transport coefficient: Mobility and Diffusion. PLASIMO requires the mobility $\mu \times N$ in units of $\text{m}^{-1}\text{V}^{-1}\text{s}^{-1}$ and the diffusion $D \times N$ in units of $\text{m}^{-1}\text{s}^{-1}$. You can convert the pressure-based units from table 5.1 and use Mobility[Constant＊mu]N. Other option is to let PLASIMO do the conversion for you. The latter can be achieved by selecting Mobility[Custom] and write the following function:

\[
0.14 \times 2 \times 10^{-6} \times 1 \times 10^{-8} \times \text{Pressure}() / (\text{Kb} \times \text{Temperature}('\text{Ar}'))
\]

In function expressions you can use the common physics constants preceded by backtick, such as $\pi$, $c$, $\text{Kb}$.

21. In Species add a section DiffusionCoefficient[Custom] and add the function:

\[
4 \times 10^{-3} \times 2 \times 10^{-6} \times 1 \times 10^{-8} \times \text{Pressure}() / (\text{Kb} \times \text{Temperature}('\text{Ar}'))
\]

22. In SpeciesList add another Species named e with initial density InitDens of $5 \times 10^{15} \text{m}^{-3}$.


24. To the electron species add Mobility[Custom] with a function:

\[
3 \times 10^{2} \times 2 \times 10^{-6} \times 1 \times 10^{-8} \times \text{Pressure}() / (\text{Kb} \times \text{Temperature}('\text{Ar}'))
\]

and DiffusionCoefficient[Custom] with a function:

\[
120 \times 2 \times 10^{-6} \times 1 \times 10^{-8} \times \text{Pressure}() / (\text{Kb} \times \text{Temperature}('\text{Ar}'))
\]

25. In ReactionList[List] add Reaction called ionization with the format $\text{Ar} + e \rightarrow \text{Ar}^+ + e + e$.

26. Add a section RateCoefficient[Custom] and write the function as it is given in table 5.1:

\[
(\text{MeanElectronEnergy()} > 5.3 \times \text{eV}) \ ? \\
8.7 \times 10^{-15} \times 1 \times 10^{-3} / (\text{MeanElectronEnergy()} / \text{eV} - 5.3) \times \exp(-4.9 / (\text{sqrt}(\text{MeanElectronEnergy()} / \text{eV} - 5.3))) \\
: 0 \times 1 \times 10^{-3} / \text{s}
\]

27. In Mixture add ElasticEnergyTransfer[Constant＊L/N] with the LossTerm set to $0.0 \text{eVs}^{-1}\text{m}^3$.

28. In order to overwrite the default boundary conditions, in section PlasmaRegion add a section BoundaryConditions and set Target the ionic species $\text{Ar}^+$. In BoundaryConditions add a subsection BndConfig and add two leafs ApplyAt and fill the names of the electrodes: grounded and powered in each leaf. Inside BndConfig add the desired boundary condition type: BndCond[HomNeumann].

29. Add another section BoundaryConditions with Target e. Add again the subsection BndConfig and fill the two leafs ApplyAt again for both electrodes. Inside BndConfig add boundary condition type: BndCond[HomDirichlet].

30. Repeat the above with Target ElectronEnergy.

31. In Model add DataSchedule and inside the subsection DataFiles add DataFormat[Plain].

32. In Model add MainIterConfig and inside of that add TimeSchedule.
33. In \texttt{TimeSchedule} add section \texttt{ScheduleBlock}. Set \texttt{BlockDuration} to 140 \(\mu s\), \texttt{InitialTimeStep} to 1 ns, \texttt{NumberOfStartingSteps} to 1, \texttt{dt\_min} to 1 ps, \texttt{dt\_max} to 1 ns, and \texttt{dt\_log} to 140 \(\mu s\). The right-hand side pull-down menu can be used to select more convenient units.

34. In \texttt{Model} add \texttt{MatrixSolver[SuperLU Solver]} and let the default matrix pre-ordering.

35. In \texttt{Model} add \texttt{EM[DriftDiffusion]} and set \texttt{RegionId} to \texttt{interior}.

36. In \texttt{Model}→\texttt{EM}→\texttt{EMScheduleBlock} add \texttt{Potential[Constant]} and \texttt{Potential[Sinusoidal]}. Note that the order of the Potentials in this section must match the order of the electrode materials in the \texttt{MaterialList}!

37. Set the constant potential to 0 V and in the \texttt{Sinusoidal} set \texttt{Amplitude} to 40 V and the frequency at 13.56 MHz.

38. Save the file as “model1.gum” (press “OK” to leave “Include” sections as is).

39. Install and run the model.

5.2.5 Building a 2D Drift-Diffusion Model Using BOLSIG+

Model specifications

This model is a 2D cylindrical DC helium discharge in a quartz tube. The tube is 30 cm long, has a diameter of 4 cm, and on opposite ends of the tube two electrodes are placed over which a voltage of -400 V is applied. Again we will begin with creating the computational grid, followed by the species and their reactions, and finally the control parameters.

In this example, we will see how to specify transport and reaction rate coefficients in a form of lookup tables. For the mobility of helium ions we will use a lookup table as a function of the reduced electric field from [8]. For the electron transport coefficients and electron-related reaction rate coefficient we will use lookup tables as a function of the mean electron energy. These can be generated externally from the free-ware Boltzmann solver BOLSIG+ \url{http://www.bolsig.laplace.univ-tlse.fr/} and included in PLASIMO. BOLSIG+ provides an option of saving the output data in PLASIMO format and the tables can be directly included in PLASIMO without any manual manipulation.

Model construction

1. Start PLASIMO as described in section 2.2.
2. Click \texttt{create new model} and select \texttt{DriftDiffusion-2D}.
3. In \texttt{Model} add \texttt{Grid[cylindrical]}. You can give a name of this grid, for example “mygrid”.
4. Set \texttt{z\_max} to 0.3 m and \texttt{r\_max} to 0.02 m.
5. In \texttt{Grid[cylindrical]} add subsection \texttt{Geometry} and set \texttt{BaseRefinementPower} to 4 2 (note the space in-between the numbers). The two numbers specify the \texttt{BaseRefinementPower} for the first and second direction, respectively.
6. In section \texttt{Grid[cylindrical]} add first a subsection \texttt{Stretch[OnePt\_Stretch]} and specify Point 0.0 and Density 2.0. This mean that stretching will be applied close to the left boundary, i.e. the cathode. Secondly add another section \texttt{Stretch[No\_Stretch]}. Note that the first \texttt{Stretch} section applies for the first coordinate (in this case \(z\)), and the second section for the second coordinate (r).
7. In \texttt{Geometry} add 5 \texttt{Material} sections, named plasma, cathode, anode, quartz, and boundary, with the respective \texttt{Id} values 0 through 4.
8. In section Material with a name plasma add a leaf Property by right-clicking in the Property frame. In this leaf enter the key-value pair Type — DischargeGas.

9. In both the cathode and anode material add a Property key-value pair Type — Electrode.

10. In the quartz material add the Property two key-value pairs: Type — Dielectricum and Eps_r — 6.0.

11. In the boundary material add a Property key-value pair Type — Boundary.

12. In Geometry add section CellMatrix. The default cell matrix contains already the four materials. The black color represents the plasma material surrounded by the other materials from each side. Change the cell matrix as follows:

   (a) Select material boundary from the pop-up menu (right-click on the grid).
   (b) Set the top and bottom rows to the material boundary.
   (c) Select material quartz and set the second row from the top to the material quartz.
   (d) At this point your model in progress should look like the one in figure 5.2 (the cell colors may be different).

![Figure 5.2: The cell matrix with assigned materials. The black color represents the plasma with Id 0, The blue on the left is the cathode (Id 1), cyan the anode (Id 2), purple the dielectric (Id 3) and the white is the boundary (Id 4). The axes of symmetry is on the bottom of the cell matrix.](image)

13. In Grid→Geometry add one Region section. Set the name of this region to plasma and enter 0 in Involves. PLASIMO set first a region called interior that contains all the materials in-between the boundaries, in this case the interior region contains materials plasma and dielectric. By adding a region plasma we instruct PLASIMO to create one more region. The EM calculations will be done on region interior, while the plasma calculations will be done on the second region.

15. Set RegionId to plasma and set Pressure to 1 Torr.

16. In PlasmaRegion add GasTemperature[Fixed] and inside TemperatureFunction set the temperature to 300 K.

17. In PlasmaRegion[DriftDiffusion] add a subsection Mixture[DriftDiffusion]. The sections is created with 3 subsections: GasList, SpeciesList, ReactionList[List]. We will continue with filling in these sections.

18. Section GasList
   - Add a section Gas and set Name to He.
   - In Gas add State[Atom] and set its Name to ground.

19. Section SpeciesList
   - Add Species and set name to He^+.
   - Set InitDens to 5e13*m^-3.
   - Add Species → Mobility[muN(E/N)]. Enable the leaf File and select the file containing the mobility LookupTable: plmd2d_v2/input/data/mobility_ions/mobility_HeinHe.lut.
   - Add DiffusionCoefficient[Einstein].
   - Add a second Species section with Name e, and set the InitDens to 1e15*m^-3.
   - Add a State[Atom] named electron with Degeneracy 2.
   - Add Mobility[muN(eps)]. Enable the leaf File and select the file containing the mobility LookupTable: plmd2d_v2/input/data/bolsigdata_He/mobility.lut.
   - Add DiffusionCoefficient[Einstein].

20. Section ReactionList[List]
   - Add Reaction, call it ionization and set the format to He + e -> He^+ + e + e.
   - In Reaction add Rate[K(eps)] and select the file plmd2d_v2/input/data/bolsigdata_He/C3_He_Ionization_24.58_eV.lut.

21. In Mixture add ElasticEnergyTransfer[EnergyLoss/N(eps)], enable the leaf File and select the file plmd2d_v2/input/data/bolsigdata_He/elastic_power.lut.

22. In section PlasmaRegion add section WallProcessList. In this section you can add processes such as Reflection, Secondary emission, Sputtering, Thermoionic emission, Photoemission. In this example we will add Secondary emission process. Add a WallProcess of type SecondaryEmission and fill in the following: Projectile He^+, SecondaryEmissionCoefficient 0.2 and SecondaryEmissionEnergy 2 eV. Add two leaves ApplyAt and add the electrode's names: cathode and anode.

23. In Model add EM[DriftDiffusion] and set RegionId to interior.


25. In the first set V0 to −400 V, and in the second to 0 V. Note that the order of these specification correspond to the order of the materials in the MaterialList, i.e. the first section Potential will apply the specified potential to the first electrode that appears in the list of materials.
26. In `Model` add `MainIterConfig`.

27. In `MainIterConfig` add `TimeSchedule`.

28. In `MainIterConfig`→`TimeSchedule`→`ScheduleBlock` set `BlockDuration` to `5e-5*s`, `InitialTimeStep` to `10*ps`, `NumberOfStartingSteps` to `1`, `dt_min` to `1*ps`, `dt_max` to `10*ns`, `dt_log` to `5e-5*s`. The right-hand side pull-down menu can be used to select more convenient units.


30. Save the file as “model2.gum” (press “OK” to leave “Include” sections as is).

31. Install and run the model.

5.3 Global Models

5.3.1 Introduction

Whereas many types of models perform calculations on some sort of grid, giving spatially resolved results, be it in a single dimension or more, a Global Model instead models some volume by only calculating volume averaged quantities. It is therefore a type of Zero-Dimensional Model (ZDM), though this does not mean that the goal is to model a point in space, but rather some volume as a whole where no transport inside that volume is taken into account.

The PLASIMO Global Model calculates the densities of species and reaction rates of processes between these species, as function of time. Additionally, the electron energy density balance is solved. Apart from defining a chemistry, i.e., a set of species and their properties, and reactions occurring between these species, the user can define any form of time dependent input power. The reactions are processes that take place in the bulk of the plasma, not involving, in any way, any interaction with the surroundings. However, in many cases the user will want to create a plasma model that does account for certain processes involving something outside the chosen volume. Examples are processes like diffusion, flow, wall recombination and sputtering. Essentially these are processes that represent some sort of transport into or out of the volume. For including processes falling in the last category extra functionality is provided.

5.3.2 Example Global Model

Example input file: `gm_Ar.gum`

Location: `plplugins/globalmodel/input/`

Model specifications

This simple argon model is inspired by a model by Ashida et al. [9]. The main parts of the input file are the following sections:

1. `Mixture`→`SpeciesList`  
   The model contains three argon species (ground, excited, ion) and the electron species. The definition of the species is rather straightforward, and is mostly characterized by defining their unique name (by which they can be used in reactions) and energy. The energy is used to determine source terms for the electron energy balance.

2. `Mixture`→`ReactionList`  
   Reactions are defined by a `Format` for the reaction formula and a rate coefficient. Rate coefficients can be defined in many ways, such as Arrhenius rate coefficients and look-up tables. In this particular input file only rate coefficients of the type `Custom` are used. This is the most flexible type, since it accepts any mathematical expression. For increased flexibility certain functions are
available to use species properties in the rate coefficients, namely \textit{Temperature()}, \textit{Density}, \textit{Mass()}, \textit{Charge()}, and \textit{Weight()}. In addition, the Global Model plugin allows the user to define constants and functions that can subsequently be used in expressions in \textit{Custom} type rate coefficients. This will be explained in more detail in the section \textit{Declares}. The \textit{ApplyDB} option indicates whether detailed balancing needs to be applied to automatically include the reverse process. This will not result in an extra reaction, but only in a modified rate.

The processes included in the reactions are (de)excitation, ionization, and radiation, the rates of which are adopted from the original model.

The process of deexcitation at the wall is in this model included as a reaction, named \texttt{wall\_Ar*}. It has the same format as the radiative decay process (\texttt{Ar*\_rad}) but with a different rate coefficient. For this particular wall process this is possible, since the resulting reaction follows conservation of mass, charge, and elemental density, which are all required for PLASIMO reactions. The process of recombination at the wall cannot be written as a reaction, since the required electrons are drawn from the vessel wall. To include such a process so-called \textit{Extra Sources} can be used.

3. \textit{ElasticEnergyTransfer}

This section is intended specifically for including energy exchange processes between heavy particles and electrons. Only the name, rate, heavy species involved in the exchange are needed. The amount of energy that is exchanged per event is

\[
\frac{2\mu_{eh}}{m_e + m_h} \frac{3}{2} k_B (T_e - T_h),
\]

with \(\mu_{eh} = \frac{m_e m_h}{m_e + m_h}\).

4. \textit{ExtraSourceList}

To complete the model the process of recombination at the wall must be included. This is accomplished by defining two \textit{extra sources}: one for the density and one for the energy. The \textit{DensitySource} takes into account the sink of ions and the source of neutral ground atoms, i.e., the \texttt{Target} leaves with the appropriate stoichiometric coefficients. Effectively, this represents the reaction \texttt{Ar\_+} \rightarrow \texttt{Ar}. The \texttt{Source} leaf represents the density with which the rate coefficient is multiplied in order to obtain the rate. This physical process also results in loss of energy to the wall, which is accounted for by an \textit{EnergySource}. For this extra source the stoichiometric coefficient needs to be defined (the target is implicitly the electron energy) as well as the energy that is transferred per event.

Using extra sources it is possible to include any kind of process in the global model, since every physical process in this kind of model is represented by a frequency of species particles and energy either appearing into or disappearing from the plasma volume.

5. \textit{Declares}

The rate coefficients for processes can become quite complex. An example of this is the rate coefficient for ion recombination at the wall

\[
K_{\text{recomb,Ar}\_+} = \frac{A_{\text{eff}}}{V} u_B, \quad (5.12)
\]

which in the model by Ashida when completely written out is

\[
K_{\text{recomb,Ar}\_+} = \frac{1}{\pi R^2 L} \left[ 0.86 \left( 3.0 + \frac{L}{2 n_{\text{neutral}} \sigma_i} \right)^{-\frac{1}{2}} 2\pi R^2 + 0.80 \left( 4.0 + \frac{R}{n_{\text{neutral}} \sigma_i} \right)^{-\frac{1}{2}} 2\pi RL \right] \sqrt{\frac{k_B T_e}{M_{\text{Ar}\_+}}},
\]

where \(R\) and \(L\) are the radius and length of the cylindrical vessel and \(\sigma_i\) is the ion neutral collision cross section. As an expression in the Global Model input file this would be something like
which is not very practical and error-prone. To make this more readable and manageable the user can define constants and functions, which can be used in rate coefficients or in other constants and functions, and so on. The difference between a constant and a function is that a constant will not change during the course of the model, while a function can. Additionally, a function can have parameters, namely zero, one, two, or three, i.e., a nullary, unary, binary, or ternary function, respectively. In gm_Ar.gum this feature is used to first define a number of constants representing the geometry of the vessel (radius, length, areas, and volume), followed by a number of functions representing quantities that might change during the course of the model, such as the diffusion length, and Bohm velocity. The Bohm velocity function also takes as a parameter a species name so it can be applied to different species. Note that nullary functions are functions, so when used they must be accompanied by a set of empty parentheses. Also note that in principle a function can be used in the definition of a constant, but since the result is a constant it will not change and be fixed at the value at which it is initially evaluated. Furthermore, any of the definitions in Declares can be used in any of the rate coefficients (as long as they are of the type CustomRate). The definitions and their composition in this example are completely optional. For instance, the length is a constant, but it might also depend on some other quantity and therefore change. Finally, there is one variable called time which represent the current time of the model in seconds. Though it will change during the course of the model it is written as a constant, i.e., without parentheses, but it acts as a nullary function.

6. InputPowerDensity
The power density input into the system must be defined in a separate section. This must always be an expression that evaluates to W/m³ or compatible unit. In gm_Ar.gum a trick is used to include an input power from a look-up table. The mathparser allows for functions to be defined which are evaluated as a look-up from a table in an external file. In this case the file gm_power_modulation.lut contains the power as function of time. By using the built-in time variable as index into the table and dividing the result by the Volume (see the appropriate Declare), an input power density as function of time is defined. Its shape is whatever the table looks like, using linear interpolation.

7. InitialValues
This section contains the initial densities of all the species present in the system. If a species is not mentioned in this section its initial value is zero.

8. Schedule
This section simply states the start and time of the model. Note that it is not compulsory for the model to start at time zero.

9. Stepper
The stepper is the ode solver that is to be used. Depending on the solver some additional options are available, such as maximum time step, tolerance, etc.

10. Options
Some general options for the model can be set. By default, for each defined species a balance equation will be constructed and solved. By adding a species to the ConstantDensity section the source term for that species will be set to zero so it will remain at its initial value. Furthermore, quasi-neutrality can be imposed, so the electron density is not solved separately, but set equal to the total ion density. By default the electron density will equal the summed ion density:

\[ \sum_i (q_i n_i) - n_e = 0, \]  \hspace{1cm} (5.13)
where the sum runs over all ions $i$, with each having a charge $q_i$.

Since it is possible to define ion species that are not meant to be part of the bulk plasma (a pseudo species at the wall for instance), the user can optionally define which ion species are to be used to calculate quasi-neutrality.

Finally, a minimum value of the electron density can be imposed. If this is enabled the electron temperature will not fall below the heavy particle temperature.

11. **Output**

Finally, the location, extension, and filenames for the output are required. Only a file name for the densities and temperature (combined in one file which also contains the energy density) is required, the output of rates, sources, and sinks are optional.

5.3.3 Building a Global Model from Scratch

1. Start plasimo as described in section 2.2.

2. Create a new model (using the menu or toolbar) of the type GlobalModel.

3. Add a Mixture section.

4. In the automatically created Species section inside the SpeciesList section that is already there set the Name to Ar.

5. Add to this particle a State[Atom] named ground.

6. To SpeciesList add another Species named Ar+ and add to it a section State[Atom] called ion with Energy set to 15.76 eV and Weight to 6.

7. Add a third Species called e and add to it a section State[Atom] named ground with Weight set to 2.

8. In the section ReactionList inside the Mixture section, add a Reaction and give it the name ionization and set the format to e + Ar -> e + e + Ar+

9. In the newly created Reaction section add a section RateCoefficient[FromCrossSection] and set TRange from 300 K to 100 eVT and add a section CrossSection[Step].

10. In Model add a section Declarations. Using Declare leafs, you can declare constants and functions that you can use in expressions elsewhere in the model.

11. Inside the Declarations section add a Declare[Constant] with the name Length and value $20*cm$.

12. Add another Declare[Constant] with the name Radius and value $5*cm$.

13. Add a third Declare[Constant] with the name Volume. The previously defined length and width can now be used to define the volume by setting value to `$\pi*Radius^2*Length` (note the “backtick” in `π` which is a predefined constant).

14. Add a fourth Declare[Constant] named PowerFreq with the value $2.5e5*Hz$.

15. To the Model section, add InputPowerDensity and set Function to

```
1*kW*cos(rad*time*PowerFreq)^2/Volume
```

In this expression rad is the radian unit and time is a builtin variable representing the current time of the model in seconds.
16. In the Model section, add a section ExtraSourceList, and inside the new section add a section Process.

17. In the Process set the name to wall_recomb. This term represents recombination at the wall, i.e., ions leaving the volume and neutral ions entering the volume. Set the Format of the reaction: \( \text{Ar}^+ \rightarrow \text{Ar} \). Set the Stoichiometry to \(-1\) (it is a loss term), and set the expression for the electron energy loss at the wall as \( 5.0/2.0 \times k_B \times \text{Temperature}'e' \).

18. Add a Rate[Constant] to the Process[wall_recomb] section and specify value for the rate coefficient equal to \(1e5\text{Hz} \).

19. To the Model section add a section InitialValues[Values].

20. Inside section InitialValues[Values] add section Densities and add three Species leaves, one for each species and fill in the species names and set the density of \( \text{Ar} \) to \(1e20\text{m}^{-3} \) and of \( \text{Ar}^+ \) and \( e \) to \(1e17\text{m}^{-3} \).

21. To the Model add a section Temperature of type 2T.

22. Inside section Temperature[2T] add a subsection GasTemperature of type Constant and set the value to 300 K.

23. Inside section Temperature[2T] add a subsection ElectronTemperature of type Calculated and set the InitialValue to 300 K.

24. In the Model section, add a section Schedule, and set EndTime to \(4e^{-4}\text{s} \).

25. In the Model section, add a section Options.

26. In the Model section, add a section Stepper[ODEPack_LSODA].

27. In the Model section, add a section Output.

28. Save the file as “mymodel.gum” for example (press “OK” to leave “Include” sections as is).

29. Install and run the model.

### 5.3.4 Solver Issues

There are several issues that can prevent the model from obtaining a result. The first issue has to do with the physics that are being modeled. In some cases it is possible for the model to reach a non-physical state, meaning that negative particle or energy densities are being calculated. The cause of this can be that there is no compensation included in the model for a loss mechanism, or that the compensation is too low. Strictly speaking, the global model has no built-in functionality to detect or prevent such imbalances in the mechanisms. A milder form of this case is when densities or the electron temperature reach such a low value that they do not represent a realistic value. The global model has two settings that can help in these cases, though their effectiveness is limited and not guaranteed. These options are UseTeFloor and MinimumDensity under Options. What these options do (when enabled) is that when the source term is calculated for the ODE system, the corrected value for either the electron temperature or any of the densities is used instead of the actual value. So this does not change any of the actual values in the model, but only uses a correction for calculating the source term.

The second issue is of a more numerical nature. This is typically the case when the source term of a species is negative, but also depends on the density of that same species. The expected result is that the species density will go down to zero but will never actually become zero because the negative source term then also becomes zero. However, because the step of the solver might be too large the result after that step might actually be negative. Some of the solvers allow for an absolute tolerance to be set which in this case must be set to a lower value, so that the solver will take smaller steps to prevent it from coming up with negative values.
A third issue is that sometimes the solver will not be able to even calculate the first step. This typically happens when there is a too large difference between the initial densities of the different species. Though it might be tempting to set the densities to zero for any species except the background or feed gas species, it is usually not strictly necessary, since also in reality densities of ions, excited species, and radicals have a low but non-zero value because of external influences like cosmic radiation. In practice a relative density difference of 12 orders of magnitude is a reasonable starting condition. Another option is to use the relative tolerance setting some solvers provide. Setting this to a low enough value can allow for a larger relative difference between initial densities.

5.4 Heat Transport Models

5.4.1 Introduction

TBD

5.4.2 Transient Heat Conduction in a One-dimensional Medium

Example input file: cartesian_1d_transient.gum
Location: plplugins/temperature/input/

Statement of the physical problem

Consider a continuous one-dimensional Cartesian geometry in which heat conduction takes place. In the absence of heat sources the temperature $T(x,t)$ is governed by the equation

$$\frac{\partial \rho c_p T}{\partial t} - \frac{\partial}{\partial x} \left( \lambda \frac{\partial T}{\partial x} \right) = 0. \quad (5.14)$$

Here $\rho$ is the mass density of the medium, $c_p$ its specific heat and $\lambda$ its thermal conductivity. If these coefficients are constant in time and uniform in space, this equation can be written as

$$\frac{\partial T}{\partial t} - \alpha \frac{\partial^2 T}{\partial x^2} = 0, \quad (5.15)$$

where

$$\alpha = \frac{\lambda}{\rho c_p}. \quad (5.16)$$

Let us assume that for $t < 0$ the temperature is uniform and given by $T = T_i$. At $t = 0$ the temperature at $x = 0$ is suddenly increased to the value $T_w$. Because of heat conduction the surrounding medium will gradually attain the temperature $T_w$ as well. If the medium is unbounded, the solution for $x \geq 0$ is given by the analytical expression

$$T_a(x,t) = T_w + (T_i - T_w) \text{erf} \left( \frac{x}{2\sqrt{\alpha t}} \right). \quad (5.17)$$

where we have used the error function $\text{erf}(u)$, which is defined as

$$\text{erf}(u) = \frac{2}{\sqrt{\pi}} \int_0^u e^{-u^2} du. \quad (5.18)$$
Numerical simulation

In order to obtain the solution by means of numerical simulation, we will restrict ourselves to the spatial interval \( x \in [0, x_0] \) and to times \( t \in [0, t_0] \). The initial condition of the problem above can be stated as

\[
T(x, 0) = \begin{cases} 
T_w & x = 0; \\
T_i & x \in (0, x_0].
\end{cases}
\] (5.19)

The boundary conditions need a little more thought. At position \( x = 0 \) the boundary condition is straightforward: the temperature is given by \( T(0, t) = T_w \). But at the boundary \( x = x_0 \) there is no natural boundary condition: in order to get the same result as for the unbounded case, we impose a boundary value in accordance with the analytical solution,

\[
T(x_0, t) = T_a(x_0, t).
\] (5.20)

With this choice the solution of the boundary value problem is the same as that of the original problem, which was defined on the half-open interval \( x = [0, \infty) \), and we expect that the numerical solution will be an approximation of this result.

Additional notes

We can use the simulation parameters \( x_0 \) and \( t_0 \) as characteristics length and time scales and use those to write the governing equation in terms of dimensionless parameters \( \tilde{t} = t/t_0 \) and \( \tilde{x} = x/x_0 \). Is is readily established that the result can be written as

\[
\frac{\partial T}{\partial \tilde{t}} - Fo \frac{\partial^2 T}{\partial \tilde{x}^2} = 0.
\] (5.21)

Here we have introduced the dimensionless Fourier number \( Fo \), which is defined as

\[
Fo = \frac{\alpha t_0}{x_0} = \frac{\lambda}{\rho c_p x_0^2}.
\] (5.22)

The Fourier number characterizes the degree of temperature leveling that is achieved by conduction in the medium with size \( x_0 \) in the time interval \( t_0 \). If the Fourier number exceeds unity by far, the temperature will be nearly uniform at \( t = t_0 \), for a small Fourier number the temperature at \( x_0 \) will still be close to its original value. The analytical solution in terms of \( \tilde{t}, \tilde{x} \) and the Fourier number is given by

\[
T(x,t) = T_w + (T_i - T_w) \text{erf} \left( \frac{\tilde{x}}{2 \sqrt{Fo \tilde{t}}} \right).
\] (5.23)

A fully dimensionless form of the solution can be obtained by defining the dimensionless temperature \( \tilde{T} \) as

\[
\tilde{T}(\tilde{x}, \tilde{t}) = \frac{T(x(\tilde{x}), t(\tilde{t})) - T_w}{T_i - T_w},
\] (5.24)

for which we find

\[
\tilde{T}(\tilde{x}, \tilde{t}) = \text{erf} \left( \tilde{x}/ \left( 2 \sqrt{Fo \tilde{t}} \right) \right).
\] (5.25)

5.5 Non-LTE Models

5.5.1 Introduction

The PLASIMO non-LTE model describes a quasi-neutral plasma. The unique feature of this model is that the diffusive velocities are calculated self-consistently from the Stefan-Maxwell equations. This set of equations is derived from a simplified set of species momentum equations. Simpler models use Fick’s law to calculate the diffusive velocities. Such an assumption is only correct when there is a dominant
background gas that is responsible for most of the interactions with other particles. In plasmas this assumption is easily violated due to the much larger cross sections of charged-charged interactions in comparison to neutral-neutral interactions. Additionally, there may be strong gradients of the species densities that do not guarantee a dominant background gas everywhere in the plasma.

In a 2D or 3D model the mass averaged velocity is solved for as well. The calculated species diffusive velocities are inserted in the species mass balances. These mass balances account for the temporal behavior of the species mass densities, the diffusive and convective transport, chemical reactions and radiative transitions.

The energy balance consists out of the two-temperature equations. It is assumed that the heavy-particles can be described by a single temperature, $T_h$. The electrons can have a deviating temperature, $T_e$.

### 5.5.2 Cascaded Arc Model

**Example input file:** arcargon.gum  
**Location:** plplugins/2T_ambipolar/input/

**Model specifications**

This model deals with a plasma created in an arc by means of a DC current. The plasma source has the shape of a cigarette: it is a cylinder with a radius of about 2 mm and a length of about 60 mm. The current creating this plasma is about 50 A. We write “about” because you can play with these plasma control parameters and look what effect they have on the plasma properties, like the electron density and temperature ($n_e$, $T_e$), the gas temperature ($T_g$), and the flow field (velocities).

After loading the model, it can be explored by navigating through the leaves in the Model editor. The main sections that are present are:

- **MainIterConfig**: This section contains some parameters to control the iterations.
- **PlasmaRegion**: The input data is organized in the following subsections:
  - **Grid**: contains specifications of the geometry, coordinates and involved materials.
  - **TemperatureHP**: This section describes the heavy particle temperature. It contains a separate boundary condition (BndConfig) for every wall, and control of the iterations is handled in the IterConfig subsections.
  - **TemperatureEL**: Similar to TemperatureHP only now for the electron temperature.
  - **Mixture**: This describes the composition of the plasma, i.e. the species that are taken into account in the simulation and the relevant reactions that they are involved in.
  - **Flow**: This section configures the flow field calculations.
- **EM**: The EM calculations. In this case it is an uniform current of 50 A.

**Results and discussion**

After installing the model in the Model data tab, you can see the run time plots and monitor the variables. All the calculated variables are organized in sections. The plots can be open from the tree structure from the left panel.

- **General**: gives the status of the calculation and the convergence log graph. During run time the graph gives the residue as a function of the iteration number;
- **plasma**: the left panel shows a tree structure containing a host of variables that are calculated by the model, organized in several groups: Barycentric, EM, Electron Temperature, Gas Temperature Mixture, Radiation and Extra Output.
The characteristics of this plasma source are the same as that of a positive column (PC). For this PC a global model can be made, which will have the general outcome:

- The electron temperature is (almost) independent of the power (density), but is dictated by the electron loss frequency. The latter is often “diffusive”, meaning that this frequency more or less equals $D_a/R^2$, where $D_a$ is the ambipolar diffusion coefficient, and $R$ the plasma radius.
- The electron density ($m^{-3}$) is determined by the power density ($W/m^{-3}$).
- The heavy particle temperature is among others ruled by the electron density and the pressure (the plasma size...). So increasing $n_e$ leads to higher $T_h$ values.

5.5.3 Building a 1D non-LTE Quasi-neutral Model from Scratch

Example input file: 3_level_Ar.gum
Location: plplugins/2T_ambipolar/input/

Model specifications

This model is a 1D model of an Ar plasma at a pressure of 400 Pa. The model includes four different species. These are the ground state Ar, an excited state Ar$^m$, an ion Ar$^+$ and an electron. Excitation and ionization are included via Arrhenius rates. Ionization is included with detailed balancing. At the wall the excited and ionized species are assumed to deexcite or recombine. The diffusive fluxes are calculated using the Stefan-Maxwell equations. A single radiative transition is included using ray-tracing to demonstrate that a significant amount of the emitted radiation is absorbed again.

The applied power in this low-pressure Ar plasma is 40 W/m. The power is expressed in units of W/m because a 1D model assumes a height of 1 m. In order to use the correct power density for a different electrode separation (height) the input power should be divided by that height.

Model construction

In this section, we will demonstrate how you can build the model described above. In this example we will not include the radiation transport calculations.

We will begin by creating the computational grid, the EM module, electron and heavy particle temperature modules, followed by the species and their reactions.

Whenever the text says to create a new section and the section contains some text between square brackets, for instance NewSection[SomeType] the text between the brackets is the Type of the section. This means that in the sub-menu of the right-click menu the appropriate entry must be selected. At all times the help window can give you more information about the section of the model currently selected.

1. Start plasimo as described in section 2.2
2. Click Create new model (left icon, or use the menu) and select ModelNonLTEInsys-1D.
3. First we will load the necessary libraries for this model.
4. Add section MainIterConfig - the configuration of this simulation. Choose Create default section from the pop-up menu. After the default section is created, click on the section in order to modify the relevant settings. You may want to specify the output path where the output files will be written on your hard disk (Directory). The rest of the settings we will leave as default in this example.
5. Add section Discretizer[SteadyState].
6. Add section EM[EMuniformE1DPower]. Set the power to 40 W. In the relevant subsection Sigma set $\sigma$ to 500/(Ohm*m)
7. In Model add PlasmaRegion[NonLTEInsys] (the only available option for this model type). You will see that three subsections already appeared. We will come back to them later. First we will set up the geometry and the mixture.

**Section PlasmaRegion[NonLTEInsys]**

With right-mouse button add the following required sections:

- 1D!Grid[cylindrical_1d]
- Mixture[NonLTEInsys]
- Transport
- Flow[Fixed]
- TemperatureHP
- TemperatureEL
- RadTrans

We will go through these sections to setup the model.

- **1D!Grid[cylindrical_1d]**
  1. In this example we will use the default name. Fill free to name the grid according to your preference. Change \( r_{\text{max}} \) to 25 mm.
  2. Add a subsection Stretch[No_Stretch].
  3. In subsection Geometry set BaseRefinementPower to 5.
  4. We need to create a list of materials that will be involved. In Geometry add Material sections named discharge, axes and glass. Set the Id in each of the materials starting with 0 for the discharge, 1 for the axes and 2 for the glass.
  5. In subsection CellVector(already created by default) you will see the default: 102. These correspond to the IDs of the defined materials. That means we have on the left hand side axes with ID 1, on the right hand side glass with ID 2 and the discharge with ID 0 is in-between. Once the list of materials is defined you can select a material and “draw” your geometry. (For information how to use the grid editor, please refer to chapter [3].)

- **Mixture[NonLTEInsys]** With right-mouse button add the following required subsections:
  - IterConfig
  - SpeciesList
  - InitFraction
  - ReactionList[List]
  - BoundaryConditions
  - Radiation
  - CrossSections
  - DiffusiveFluxModel[Ambipolar]

  1. **IterConfig**
     Add a subsection MatrixSolver[SuperLUSolver]. Set the under-relaxation factor URF to 0.1.
  2. **SpeciesList**
     We have to create a list of all species that will be involved in the simulation. One subsection is already predefined.
     (a) In subsection Species set the name to \( \text{Ar} \) and provide a radius of \( 1.92\times10^{-10} \) m. For this species add a subsection State[Atom]. In State set Name to \( \text{Ar\_ground\_state} \).
     (b) In order to add other species repeat the item above for every species. Add species \( \text{e, Ar}^+ \) with energy 15.759 eV and \( \text{Ar[m]} \) with energy 11.5 eV.
  3. **InitFraction**
     Add the following leaves:
Species Ar Value 0.999979
Species Ar[m] Value 1e-6
Species Ar+ Value 1e-5
Species e Value 1e-5

4. ReactionList[List]

(a) In subsection Reaction give a name of the first reaction, for instance: Ar_excitation and give the correct format for Ar excitation: Ar + e -> Ar[m] + e. Then Add a subsection RateCoefficient[Arrhenius]. Give the following values: for the constants C 8e-17*m^3/s and for the exponent TExp 0.5.

(b) Repeat the procedure above to add the following reactions with the following input data:

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>RateCoefficient[Arrhenius]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ar_ionization</td>
<td>Ar + e -&gt; Ar+ + e + e</td>
<td>C 2.61348e-17*m^3/s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TExp 0.6851</td>
</tr>
<tr>
<td>Ar_ladder_ionization</td>
<td>Ar[m] + e -&gt; Ar+ + 2 e</td>
<td>C 8e-15*m^3/s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TExp 0.5</td>
</tr>
</tbody>
</table>

5. BoundaryConditions

(a) Add a subsection BndConfig. And activate leaf ApplyAt. Add: glass All. That means these boundary conditions will be applied on material 'glass' at all interfaces. In this case only east interface.

(b) Inside the BndCond add WallProcess[ThermalFlux]. Give a name of that process, for instance "recombination" and set the correct format of that process: Ar+ + e -> Ar. Then fill FluxSpecies Ar+ and Chance 1.

(c) Repeat the above for process de-excitation:

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ar[m]_deexcitation</td>
<td>Ar[m] + e -&gt; Ar + e</td>
</tr>
<tr>
<td></td>
<td>FluxSpecies Ar[m]</td>
</tr>
<tr>
<td></td>
<td>Chance 1</td>
</tr>
</tbody>
</table>

(d) We need to add BC to the axes as well: In section BoundaryConditions add BndConfig. Set: ApplyAt axes All.

6. Radiation

Add a subsection Radiation add TransitionList->Transition[AtomicRadiation]. Give a name of this transition and add the correct format of the transition: Ar[m] -> Ar. Set the number of sample points to 1000 and the frequency range to 100 GHz. For the line transition use the following: Line 0*eV 8e+06*1/s 1 1.

7. CrossSections

We will leave this section empty. If nothing is specified here PLASIMO will find the best from the data base.

8. DiffusiveFluxModel[Ambipolar]

Nothing to be done here.

9. ChemSourceClipping

Nothing to be done here.
Transport. The required sections are predefined. Inside every transport section we have to add Calculator - how this transport coefficient will be calculated. Add the following:

- ThermalCondHP -> Calculator[Thc_HPTranslational_MC]
- ThermalCondEl -> Calculator[Thc_Electron_Devoto]
- SpecificHeatEl -> Calculator[StdSpecificHeatEl]
- SpecificHeatHP -> Calculator[StdSpecificHeatHP]
- ViscosityHP -> Calculator[visc_HP_CE1]
- ElectricalCond -> Calculator[ElecCond_Devoto]
- ElasticEnergyTransfer -> Calculator[StdElasticEnergyTransfer]

Flow[Fixed]

1. In PressureFunc→Function set the pressure to 400*Pa.
2. In MassFluxDensFunc→Function set the function to 0*kg/(m^2*s). You need to add two more sections MassFluxDensFunc (for the three velocity components) and set the functions to 0*kg/(m^2*s).

TemperatureHP

1. Set the initial value InitialValue to 600 K.
2. In subsection BoundaryConditions→BndConfig fill: ApplyAt glass All. Then add BndCond[ConstDirichlet] and give a value of 300*K.
3. Add one mode section BndConfig and set: ApplyAt axes All. Then add BndCond[HomNeumann].
4. In section TemperatureHP→IterConfig add a section MatrixSolver[SuperLUSolver].

TemperatureEL

1. Set the initial value to 9500 K.
2. In section TemperatureEL→IterConfig add a section MatrixSolver[SuperLUSolver].
3. Add a section BoundaryConditions→BndConfig. We will apply homogeneous Neumann BC at the glass and the axes. Add two leaves ApplyAt: ApplyAt glass All and ApplyAt axes All. Then add a subsection BndCond[HomNeumann].

RadTrans

Add a subsection Transfer[Default].

Save the file as “model1.gum” (press “OK” to leave “Include” sections as is). Install and run the model.

5.6 LTE properties calculator

5.6.1 Introduction

The model first calculates the LTE composition according to the system of Guldberg-Waage equations. Once the composition is obtained the transport properties will be calculated. These properties include the viscosity, electrical conductivity and the thermal conductivity. Additionally, the specific heat is calculated. It is also possible to inspect some intermediate results like the Debye length or the internal partition sums.

5.6.2 LTE properties of Ar mixture

Example input file: argon_4_ion.gum

Location: plplugins/lte_properties/input/
Model details

Three sections must be specified in order to calculate the transport properties. The first section is the mixture section. The mixture section contains a SpeciesList. This example model uses species definitions from /input/mixture/species where the particle states are defined. For atoms the most important entries in the states are the energy level $E_i$ and the statistical weight, $g_i = 2J_i + 1$ with $J_i$ the total angular momentum. Note that the energy levels must be defined relative to a consistent basis. For example, in the Ar system the ground state of neutral Ar is set to 0 eV. That means that all the other energies are measured relative to the Ar ground state. This includes all Ar internal states, but also every possible Ar ion. For that reason the energy of the ground state of Ar$^+$ is not zero.

Another subsection inside Mixture is InitFraction. This section specifies the mass fraction of the elements that occur in the mixture. Note that the electrons are also considered as an element. For a neutral plasma the electron fraction should be set to zero. Since there is no other element than Ar, the Ar mass fraction is set to 1. It is recommended to not change the other subsections in the Mixture section.

The second section that must be specified is the Transport section. Specific calculators can be chosen for the various properties. It is recommended to use the settings given in the example for all calculations. Note that for the thermal conductivity four calculators are specified. These represent the four contributions that make up the thermal conductivity: The translational contributions of the heavy particles and the electrons; the reactive thermal conductivity and the thermal conductivity due to internal excitation. The last two contributions represent the transport of chemical and internal energy.

The third section is TRange. It specifies the temperature range for which the calculation will be performed. A minimum and a maximum temperature can be selected along with the number of points. Additionally you can choose between a linear or a logarithmic distribution of points with the option LogarithmicT.

By installing the model the ‘Model data’ tab appears. Inside this tab you can inspect the transport properties and the calculated composition. The results are updated whenever values are available for the next temperature point.

5.7 Other Models

5.7.1 Monte Carlo model

Example input file: mc-demo.in
Location: input/mc/

Model specifications

This model is simple kinetic drift model. We have Ar gas (density swarm) and we inject 10000 particles (electrons) into the simulation. The particles are injected as point sources at coordinates (0,0,0). An uniform electric field is applied in z-direction.

The main branches of the tree structure in the Model Editor tab are:

- **Environment**: this node contains the setup of the vessel, specifies the particles that are injected into the vessel, and defines the swarm list, i.e. the environmental particles the injected particles will collide with (the background gas). In this example vessel of type Infinity is specified. This practically means that there is no vessel. The particles are moving under the influence of a constant electric field. The electric field is applied in z-direction (pointing to the screen). There are two more “Vessel” sections which are disabled. Vessel of type plate places a wall (a plate) at some specified distance (in z-direction) from the particle injection source. You can activate this section by pressing right mouse button over the disabled section.

- **Statistics**: obviously the necessary statistics are collected in this node, such as particle positions and velocities, and EDF’s of the particle swarm that is followed. Moreover, the Statistics class
can collect statistics of wall collisions (if there is any wall), such as position and velocity at the moment when the particle hits the wall, as well as EDF and deposition profile.

- **mcSpeciesList**: This section contains a list of all species that are involved in the simulation, hence the model should know about.

- **FlightControl**: this node controls the “flight” of the particles, that must be followed, the chemical reactions that they are involved in and the time settings of the “flight”. Therefore, the FlightControl section contains SwarmList, mcProcessList and TimeSettings subsections.

After installing the model a tab Model data appears that contains all the output data. The output data is grouped into three main sections, as you can see on the left-hand side:

- **General**: containing the status of the calculation and the model-specific general data.

- **Collision frequency**: the collision frequencies for all collision processes.

- **Species**: positions, velocities and EDF for the followed particles. If specified in the input, the wall EDF or deposition profile of the followed particles will appear.

![Figure 5.3: Monte Carlo model.](image)

Few steps after starting the model you should see already the multiplication of the particles due to ionization processes. Recall that the electric field is applied in direction pointing to the screen. So that the electrons will move in opposite direction, away from the screen. To see clearly the moving of the particle cloud rotate the plot as shown in figure 5.3.

5.7.2 **EM model - 1**

**Example input file**: circular_bessel.gum

**Location**: plplugins/em/input/
The model represents transversal magnetic modes in a circular waveguide. The geometrical configuration is shown in figure 5.4.

![Figure 5.4: Geometry](image)

**Algorithm**

- Finite Differences in Frequency Domain
- Full Vector TM (Transversal Magnetic): $E_r, E_z, \tilde{H}_\phi$

**Standing waves and Evanescent Solutions**

Install the program, and run it. Plot the normalized magnetic field $\tilde{H}_\phi$.

Data/Cyl/Fields/em_region/Field (complex)/H_ud

The excitation boundary creates a wave in $z^+$-direction, that interacts with the wave reflecting back from the metal wall at the end (2). Thus, standing waves are observed.

Below a critical radius, $R_c$, there are only evanescent (decaying) solutions instead standing waves. Decrease the radius $R$, until only evanescent solutions appear in the circular waveguide.

### 5.7.3 EM model - 2

**Example input file:** radial_to_coaxial wg show.gum

**Location:** plplugins/em/input/

The model represents radial-coaxial waveguide coupling. The geometrical configuration is shown on figure 5.5.

**Algorithm**

- Finite Differences in Frequency Domain
- Full Vector TM (Transversal Magnetic): $E_r, E_z, \tilde{H}_\phi$
Conversion from Radial to Coaxial solution

Install the program, and run it. Plot the normalized magnetic field $\tilde{H}_\phi$

Data/cyl/em_region/Field(complex)/H_ud

The excitation boundary creates a wave in $r^-$-direction. When it reaches the coupling, the wave is converted into a wave propagating in $z^+$-direction. An open boundary condition (2) prevents most of the reflection, and almost no standing wave pattern is created in the coaxial waveguide. However, in the radial parallel plate waveguide, there is a standing pattern due to the reflection with the metal wire at the axis.

- Go to the Model editor and select in the hierarchical tree Model/Grid/Geometry/CellMatrix. You can now modify the geometry and draw a triangle (pyramidal cylinder in 3D), such as the one shown in the figure below.

Right click and Select Material. Choose the Metal. Change the materials cells to metal by left clicking on them. Note that you can also enable the other section CellMatrix, where the triangle is already included.

What happens to the standing waves in the radial waveguide?
• Change the boundary (2) from the open boundary condition (plane wave tangential E fields) to the homogeneous Dirichlet condition (null tangential E fields). You should notice a standing wave appearing in the coaxial waveguide.
Bibliography


