1. PySimulator

PySimulator provides a graphical user interface for simulating different model types (currently Functional Mockup Units and Modelica Models), plotting result variables and applying simulation result analysis tools like Fast Fourier Transform. Read more about the PySimulator here https://code.google.com/p/pysimulator/

1.1 Installation under Windows

In order to run PySimulator on windows you need to install the following softwares;

- Python – Install python from http://www.python.org/download/. Python2.7.3 is recommended.
- Numpy1.7.1 – Install numpy from http://sourceforge.net/projects/numpy/files/NumPy/1.7.1rc1/. numpy-1.7.1rc1-win32-superpack-python2.7.exe is recommended.
- PySide1.1.1 – Install PySide from http://releases.qt-project.org/pyside/. PySide-1.1.1qt474.win32-py2.7.exe is recommended.
- Scipy0.10.1 – Install scipy from http://sourceforge.net/projects/scipy/files/scipy/0.10.1rc2/. scipy-0.10.1rc2-win32-superpack-python2.7.exe is recommended.
- h5py – Install h5py from https://code.google.com/p/h5py/downloads/list. h5py-2.0.1.win32-py2.7.msi is recommended.
- enable4.3.0, chaco4.3.0, and traits4.3.0 – Install the all-in-one package from ETS, the Enthought Tool Suite, available also from http://www.lfd.uci.edu/~gohlke/pythonlibs/. ets-4.3.0.win32-py2.7.exe is recommended.
- python-sundials0.5 – Install python-sundials from https://code.google.com/p/python-sundials/downloads/list . python-sundials-0.5.win32-py2.7.exe is recommended.

1.2 Running PySimulator

Run the PySimulator by executing the PySimulator.py script located at OpenModelica1.9.0/share/omc/scripts/PythonInterface/PySimulator.

```
python PySimulator.py
```

This will start the PySimulator and on success you should see the output,

```
Dymola plug-in loading
FMUSimulator plug-in loading
OpenModelica plug-in loading
OMC Server is up and running at file . . .


DymolaMat plug-in loading
Mtsf plug-in loading
EigenvalueAnalysis plug-in loading
```

```
LinearSystemAnalysis plug-in loading
MinMax plug-in loading
SignalProcessing plug-in loading
```

1.3 Loading a Modelica Model

The integration of the OMPython module within the OpenModelica plugin for PySimulator makes it possible for the modeler to quickly load Modelica files such as models (.mo) or load a simulated model's executable file.

The user can open these files from the menu bar by selecting `File` > `Open Model` > `OpenModelica`. In this introductory example we will use a pre-defined model named Influenza to demonstrate the use of OMPython in PySimulator. Figure 1-1 depicts the graphical user interface of PySimulator when opening a model file. Once the model file is selected, the model is loaded into the variables browser and is ready to be configured for simulations.

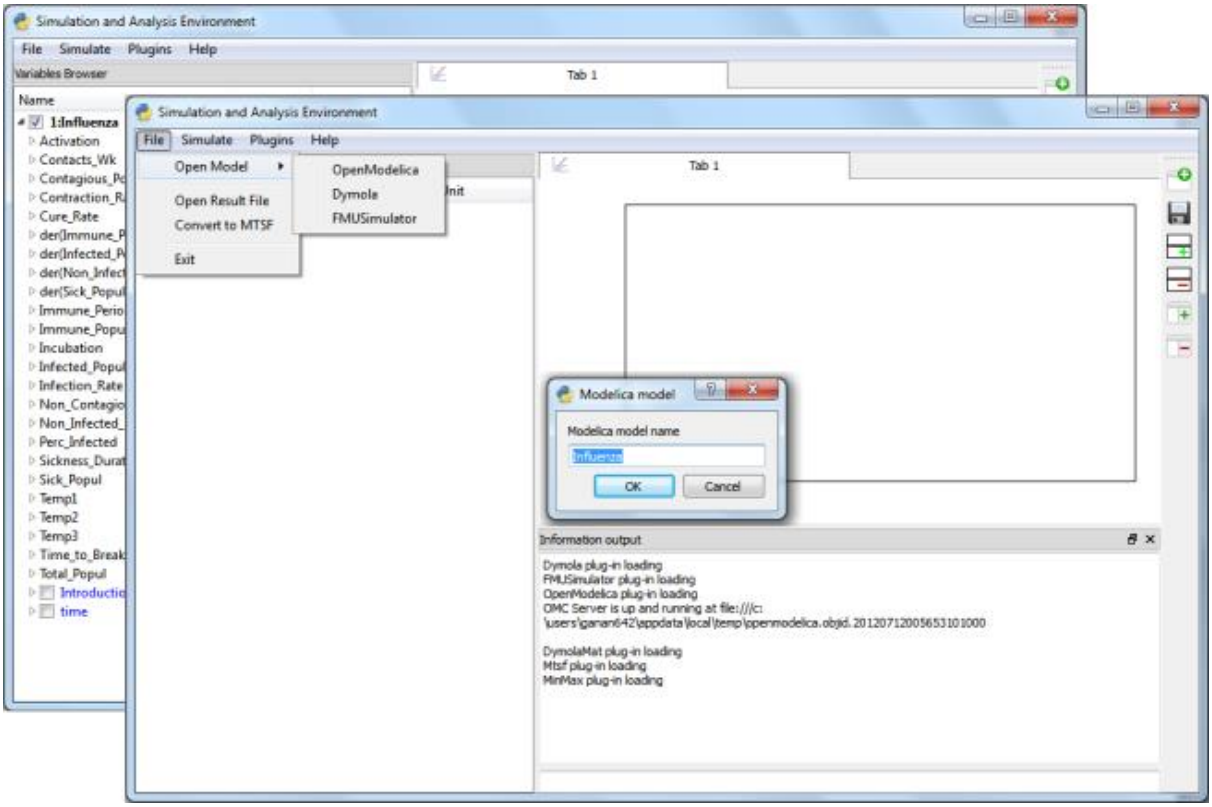

Figure 1-2: Loading Modelica models or model executables in PySimulator

1.4 Using the OpenModelica plugin

The loaded Modelica model can be simulated from PySimulator using the default simulation options or by setting the simulation options before simulating from the Integrator Control dialog box. The OpenModelica plugin defines the simulation routine for the Modelica models by using the execute method of the OMPython API.

Figure 1-2 shows how the simulation options can be set using PySimulator's Integrator control feature.

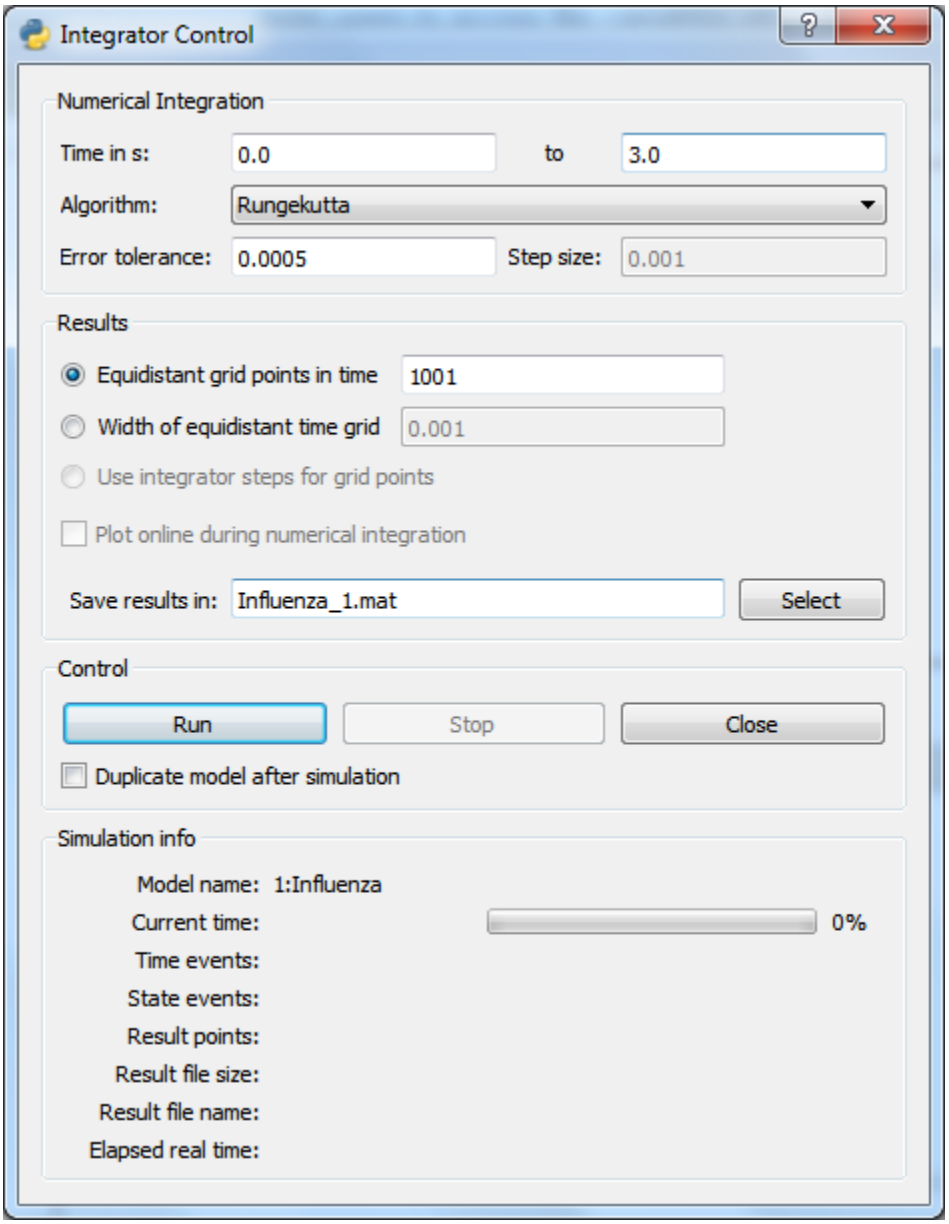

Figure 1-2. Preparing the simulation settings using the Integrator Control.

1.5 Simulating the model

The initial simulation parameters and settings are provided as inputs to the OMC via the front-end of PySimulator. The Run button of the Integrator control triggers the simulate command of the OMC with the supplied simulation options. The simulate command has the following parameters,

- Simulation Interval
  - Start Time
  - Stop Time
- Algorithm
- Error Tolerance
- Step size

The user has the option to choose from a range of Numerical integration algorithms from the Algorithm selection box. The Integrator control dialog box also filters some parameters that are not available for some integration solvers by disabling the field; avoiding error and providing more accuracy in the results.

The Variables browser builds a tree structure of the instance variables and highlights time-continuous variables in blue. The user can select these variables and plot them in the Plot window by checking the check box near the highlighted variables.

Figure 1-3 illustrates the Variables browser that allows users to access the variables after the Influenza model has been simulated with some simulation parameters set.

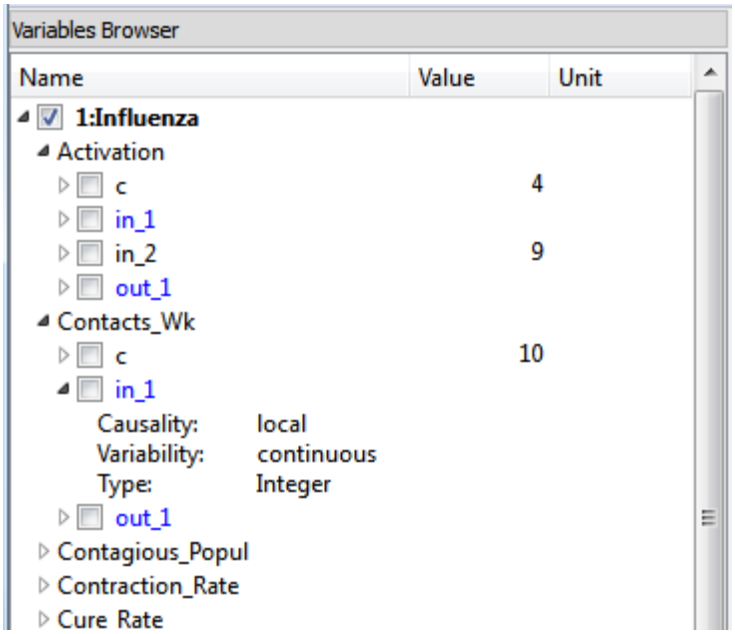

Figure 1-3. Variables browser of the simulated model.

1.6 Plotting variables from the simulated models

The Plot window of the PySimulator GUI provides additional user interface controls for comparing different plots side-by-side, adding and removing plots and also to save the plots.

Figure 8 shows the plotted variables in the plot window and the list of simulation variables in the Variables browser along with the variables selected for plotting.

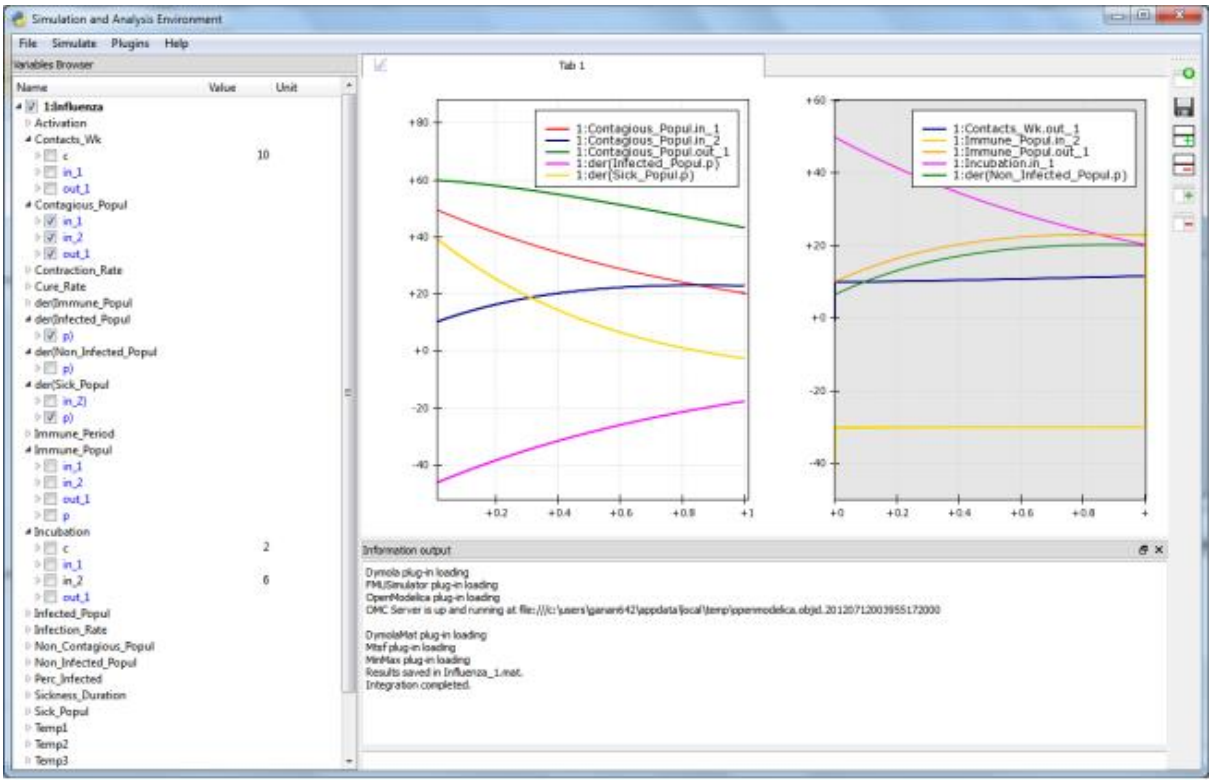

Figure 1-4. Plotted variables using PySimulator.

1.7 Using Simulated results

It is desirable to avoid simulating the model again every time the user needs the simulation results. It is instead preferable to use an existing simulation result file for the calculations, this saves resources and time. PySimulator supports opening the OpenModelica simulation result files (.mat) and the model's executable file to build the variable tree in the variables browser. The user can then adjust some parameters from the variable tree or the Integrator control to achieve the desired results.