PySHEMAT documentation

J. Florian Wellmann

September 26, 2011

Contents

1 Short overview of PySHEMAT

1.1 PySHEMAT

SHEMAT (Simulator of HEat and MAss Transport) is a coupled fluid, heat and reactive transport simulation code (Clauser and Bartels, 2003). It is widely applied in low-temperature geothermal and mineral systems simulation (e.g. Bartels et al., 2002; Kühn et al., 2006; Mottaghy and Rath, 2006; Gessner, 2009; Rühaak et al., 2010). Fluid and heat flow simulations with SHEMAT can be performed from the command line (on Windows and Linux/Unix) using an ASCII input file. The input file is structured in a Keyword-Variable notation (see Clauser and Bartels, 2003, Sec. 2.4.3 for a detailed description). Cell lithology/geology, flow properties and boundary conditions are not directly assigned to cells but stored in 1-D arrays where the first element corresponds to the lower-left cell of the model and cells are iterated from left to right and bottom to top of the model. The grid is specified by the total number of cells and the cell sizes in each coordinate direction. Results of the simulation are stored in exactly the same file format, enabling the use of the same Python code for input and output files. Simulations are limited to rectilinear (orthogonal) mesh structures, although, recently a method has been developed to enable simulation with non-orthogonal meshes, using a coordinate transformation method (Rühaak et al., 2010).

1.1.1 Methods for input file generation (Pre-processing)

The PySHEMAT library takes an object-oriented approach, representing a SHEMAT input file as an object, with all relevant variables stored in the object and specific methods of the object providing access to them. The first step is therefore either to load an existing SHEMAT .nml input file or to create an empty file from a template into a Python object:

```
S1 = Shemat_file(filename = "model.nml")
```

Model variables can easily be accessed and stored in temporary python arrays for changes. Different methods are used to read single variables (e.g. `NDIM`, the number of cells in x-direction) and array variables (e.g. `TEMP`, the cell temperatures):

```
ndim = S1.get("NDIM")
temp = S1.get_array("TEMP")
```

The variables can now be manipulated with standard Python methods and stored back in the object with the according methods, e.g. for the variables above:

```
S1.set("NDIM")
S1.set_array("TEMP")
```

For a full list of variables, see Clauser and Bartels (2003), Tab. 2.7–2.15.

The commands above can be considered as "low-level" commands, useful for direct access of the variables in the SHEMAT object. A number of "high-level" commands are also available for more complex common tasks. In many cases, a 3-D array is more useful to read or manipulate the data, instead of the 1-D array structure used in the input file. A variable can directly be obtained as a `data[x][y][z]` structure with:

```
data = S1.get_array_as_xyz_structure("TEMP")
```

or converted from a 1-D array:

```
temp_array = S1.array_to_xyz_structure(temp,idim,jdim,kdim)
```

where `idim, jdim, kdim` are the dimensions of the model. Other methods have been written to allow a change in the mesh structure, the number of cells in each direction or the positions of the mesh boundaries.

A useful method is provided to update the array variables relevant for the flow simulation (e.g. porosity, permeability, thermal conductivity) based on the geology assigned to the cells (variable `GEOLOGY` in the `.nml` file). The new values can directly be defined within the SHEMAT object or derived from an external comma-separated-value file. Any of the SHEMAT properties can be defined in this file, e.g.:

```
NAME,GEOLOGY,PERM,ANISOI,ANISOJ,POR,HPR,WLFMO
Basement,1,1.00E-19,5,5,0.01,0.00E+00,3
Permian,2,1.00E-16,5,5,0.1,0,3
Woodada_Kockatea,3,1.00E-15,5,5,0.1,0,2.5
...
```

All variables in the SHEMAT object are then automatically updated with the command

```
S1.update_properties_from_csv_list(csv_filename)
```

This method for the automatic update of all properties can be used for simple property changes, all the way to stochastic simulations for a whole range of different properties (see examples below).

With all parameters adjusted and stored back in the object, a new input file for the SHEMAT simulation can be created:

```
S1.write_file(filename)
```

Additional methods are provided to perform tasks related to running the simulation (not related to preparation of the input file). With

```
create_nml_dir()
```

a new directory is created for the simulation run, with automatic continuous numbering. The control file, essential for the SHEMAT simulation, can be created with

```
create_shemat_control_file(filename).
```

Finally, the simulation can directly be executed (if SHEMAT is installed on the system) with

```
execute_shemat()
```

Several "batch-methods" have also been implemented to simplify and automate common tasks. For example, when a new model is created, a variety of parameters and settings have to be adjusted, including, for example, the number of cells, boundary conditions and initial values for parameters like pressure and temperature. The relevant model set-up steps are integrated into a single function call, simplifying and automating the whole process. Essential arguments are the cell spacings in each direction, passed as arrays `dx,dy,dz` (cell sizes can change along each axis direction). Several other settings can be provided as optional keywords, e.g.:

```
S1_new = create_empty_model(
    dx=dx, dy=dy, dz=dz,
    title="Convection␣example",
    seitet="NFLO",
    baset="TEMP",
    topt="TEMP",
    base_temperature=40,
    compute_heat = True,
    compute_fluid = True,
    coupled_fluid_heat = True,
    initialize_temp_grad = True,
    initialize_heads = True,
    set_heads = 1000,
    nml_filename = "conv_ex_1")
```

This function provides a simple and direct way to create an empty SHEMAT model from scratch, even for complex rectilinear mesh geometries. The model can then be further adapted with the methods described above. These functions and object methods provide a wide variety of options for automatic model generation. Some simple examples are provided below.

1.1.2 Methods for output analysis and visualization (post-processing)

As pointed out before, the SHEMAT output file is in the same format as the input file, but with the file ending `.nlo`. It can, therefore, be loaded into the same object structure, e.g.:

```
S1_out = Shemat_file(filename = "model.nlo")
```

The methods that were used for input file manipulation are directly applicable to process and analyze the results. Additional functions can be used for more detailed analyses and plot generation, based on the free Python plotting library Matplotlib.

Simulated values at any point (x, y, z) in the model can directly be assessed with a simple function call, e.g.:

```
value = S1_out.get_value_xyz("TEMP",x,y,z)
```

A more complex method is available to create isohypse maps, for example the depth of the 100°C temperature isosurface in the model (interpolated between the known values at the cell centers), e.g.:

```
temp_100_2D = S1_out.get_isohypse_data("TEMP",100)
```

In many cases, a summary map for one property or for a calculated mean value is required, for example to produce a map view of the temperatures at the top or the mean temperatures of a specified geological unit. Using PySHEMAT we can, for example, calculate mean temperatures of the geological unit 2 with

```
mean_temp = S1_out.calc_mean_formation_value(2, "TEMP")
```

A simple 2-D map can be created with another function and directly saved as a figure:

```
S1_out.create_2D_property_plot(mean_temp,
   interpolation='bilinear',
   title = 'Mean␣temperatures␣unit␣2',
   colorbar = True,
   colorbar_label = 'Temperature',
   xscale = 'kilometer',
   yscale = 'kilometer',
   xlabel = 'E-W␣[km]',
   ylabel = 'N-S␣[km]',
   savefig = True,
   filename = "mean_temp.png")
```

This plotting function can, for example, also be used to create a map of calculated isosurfaces.

Further postprocessing options are implemented in the module. These include methods to export data into a grid format for Geographic Information Systems and into a voxet format for 3-D visualizations, to create slice plots through the model or histograms of simulated values.

1.2 Analysis of transient simulations

It is possible to store simulated values during transient simulations with SHEMAT at specified locations, in monitoring files. A simple Python module is available to access these monitoring files, facilitating the generation of result plots, for example with the Python package Matplotlib (e.g. **?**). The definition follows a logic similar to the SHEMAT class definition presented before (Sec. 1.1.1). The object can be initialized and the monitoring file loaded, e.g.:

```
M = Monitoring_file()
M.load_monitoring_file("model.001")
```

The data for a specific variable stored in the monitoring file, for example temperature, can directly be obtained with

```
M.get('T')
```

And a plot of this data over the course of the model simulation is created with

```
M.plot('T')
```

There are, again, a variety of “high-level” commands available to combine commonly performed tasks for the monitoring data. A very useful function exists to combine readings of a variable in several monitoring objects (stored in a list) into one plot, e.g.:

```
plot_type_in_multiple_objects(
    monitoring_object_list,
    'T',
    xlabel="Time␣[years]",
    ylabel="Temperature␣[C]",
    title="Temperature␣at␣observation␣points",
    transform_x = 1./365.,
    legend=False,
    legend_position='upper␣right',
    show=False,
    savefig=True,
    filename='temperature_dev.png')
```

References

Bartels, J., Kühn, M., Schneider, W., Clauser, C., Pape, H., Meyn, V., Lajcsak, I., 2002. Core flooding laboratory experiment validates numerical simulation of induced permeability change in reservoir sandstone. Geophysical research letters 29 (9), 1320.

Clauser, C., Bartels, J., 2003. Numerical simulation of reactive flow in hot aquifers: SHEMAT and processing SHEMAT. Springer, Berlin.

Gessner, K., 2009. Coupled Models of Brittle-plastic Deformation and Fluid Flow: Approaches, Methods, and Application to Mesoproterozoic Mineralisation at Mount Isa, Australia. Surveys in Geophysics 30 (3), 211–232.

Kühn, M., Dobert, F., Gessner, K., 2006. Numerical investigation of the effect of heterogeneous permeability distributions on free convection in the hydrothermal system at Mount Isa, Australia. Earth and Planetary Science Letters 244 (3-4), 655–671.

Mottaghy, D., Rath, V., 2006. Latent heat effects in subsurface heat transport modelling and their impact on palaeotemperature reconstructions. Geophysical Journal International 164 (1), 236–245.

Rühaak, W., Rath, V., Clauser, C., 2010. Detecting thermal anomalies within the Molasse Basin, southern Germany. Hydrogeology Journal, 1–19.