RECENT DEVELOPMENTS IN THE PYTOUGH SCRIPTING LIBRARY FOR TOUGH2 SIMULATIONS

Adrian E. Croucher[1]

[1]Department of Engineering Science, University of Auckland, New Zealand

a.croucher@auckland.ac.nz

Keywords: *Reservoir modelling, TOUGH2, Python*

ABSTRACT

PyTOUGH, a Python scripting library for automating TOUGH2 simulations, was publicly released as free open-source software in 2011. Since then, it has been used in a wide variety of TOUGH2 modelling applications, particularly for complex simulations which would be difficult or impossible using traditional manual or graphical workflows. It has also been incorporated into at least two graphical interfaces for TOUGH2.

This paper describes the considerable development PyTOUGH has undergone since its initial release. Notable new features for grid handling include generators for radial and MINC TOUGH2 grids, grid embedding and reverse engineering of rectangular grid geometry from TOUGH2 input. Support has been added for tilted geometries, auxiliary TOUGH2 mesh and time history files, and output from TOUGH+ and TOUGHREACT. New graphical features include plotting block-centred flow vectors, well tracks and rock types. In addition, PyTOUGH now offers easier installation, an enhanced user guide, a more permissive (LGPL) license and improved speed.

1. INTRODUCTION

The TOUGH2 simulator (Pruess, 2004) is widely used for modelling geothermal reservoirs, as well as other subsurface fluid and heat flow problems. The PyTOUGH library (Croucher, 2011; Wellmann et al., 2012) was released as an open-source project in 2011, making it possible to automate the pre-processing, running and post-processing of TOUGH2 simulations via Python scripting. This approach can save time and reduce the likelihood of user errors in conventional TOUGH2 modelling, and opens up a range of possibilities for more complex models which had hitherto been, in practical terms, impossible to implement.

In the four years since its initial release, PyTOUGH has undergone continued development. This paper describes the major new features which have been added over that time, and provides a brief outline of the many smaller enhancements which have also been included. Finally, some notable recent applications of PyTOUGH are indicated.

2. MAJOR NEW FEATURES

2.1 Radial and MINC TOUGH2 grid generation

TOUGH2 includes native capability for generating rectangular, radial and MINC grids via its MESHMAKER option. However, this option does not always provide the flexibility that users need. For some applications it can also entail running TOUGH2 twice, firstly to invoke MESHMAKER and secondly to run the model, with the TOUGH2 input data file having to be edited before the second run, making automation difficult.

The original PyTOUGH release already included a rectangular grid generator, and in version 1.1 a radial grid generator was added, in the form of the `t2grid.radial()` method. On top of basic radial grid generation capability it also offers flexible block naming options, as well as the ability to model fractured reservoirs using the “generalized radial flow” representation of Barker (1988), also known as a “fractional dimension” model. In this approach, the grid can be assigned any dimension between 1 and 3, including non-integer values, to represent a range of different fracture regimes around a well. The dimension only modifies the block volumes and connection areas in the TOUGH2 grid.

More recently, PyTOUGH acquired the capability to create TOUGH2 grids for fractured reservoirs based on the “Multiple INteracting Continua” (MINC) concept of Pruess and Narashimhan (1985). In this approach, additional blocks representing the rock matrix are added to the original model blocks, which now represent the fractures.

PyTOUGH's `t2grid.minc()` method offers flexible options for naming the additional blocks and rocktypes created. It can also easily perform MINC processing on only part of the model grid, leaving the remainder unmodified (useful for large grids in which MINC can become expensive if applied to the entire grid). While this can be done with MESHMAKER by changing the block order and using inactive blocks, this again makes automation difficult. Using PyTOUGH it is straightforward to automate MINC processing, for example as part of an inverse modelling process which includes MINC parameters (such as fracture spacing) in the inversion.

2.2 TOUGHREACT support

TOUGHREACT (Xu et al., 2011) is a variant of TOUGH2 which includes modelling of chemical species in non-isothermal flows. In general it uses the same input and output data formats as standard TOUGH2, and is hence implicitly supported by PyTOUGH.

However, there are some differences. The SAVE files, to which the TOUGHREACT model state is written at the end of a run, can contain permeability information as well as porosity. They also use a different format for the timing data at the bottom of the file, as the inclusion of chemical species in the simulation may necessitate much larger numbers of time steps.

TOUGHREACT can also produce additional TecPlot files containing chemical output. PyTOUGH now includes a `toughreact_tecplot` class for interacting with these files, which works similarly to the `t2listing` class for handling standard TOUGH2 output listing files.

2.3 TOUGH+ support

TOUGH+ (Moridis et al., 2008) is intended as a successor to TOUGH2, written in modern object-oriented Fortran 95/2003. However, its input files are backwards-compatible with those for TOUGH2. Additional keywords and sections may be present, and PyTOUGH does not yet support these.

TOUGH+ output files are still fixed-format text files, similar to those produced by TOUGH2, but with different formatting and with additional tables for extra quantities calculated at each grid block.. In 2012, PyTOUGH added support for TOUGH+ output in the `t2listing` class used for handling the output from other varieties of TOUGH2.

2.4 TOUGH2 history files

As well as its main listing files, TOUGH2 can optionally produce history files (named FOFT, COFT and GOFT) with time series of results at particular blocks, connections or generators. PyTOUGH now includes a `t2historyfile` class for representing these files (although they are to some extent made redundant by the `history()` method of PyTOUGH's `t2listing` class).

PyTOUGH supports the history files produced by all varieties of TOUGH2, even though the TOUGH+ and multi-processor TOUGH2-MP varieties both work quite differently.

2.5 Reverse-engineering rectangular grid geometry

The grid in a TOUGH2 data file does not contain information about any underlying geometrical grid, i.e. positions and connectivities of block vertices in space. For grid preparation and model post-processing, such auxiliary geometry information is often needed.

PyTOUGH's `mulgrid` class supports grid geometry described by MULgraph geometry files (O'Sullivan and Bullivant, 1995). This class can be used for a wide variety of geometric grid operations, e.g. grid generation, rotation, refinement and surface fitting, as well as post-processing. However, this is of little use to users who have created TOUGH2 grids using other means and do not have a MULgraph geometry file available.

For a general unstructured TOUGH2 grid, it is not possible to reverse-engineer a geometric description. However, it is possible for rectangular grids, and as such grids are commonly used, this is a useful exception.

In version 1.4, PyTOUGH's `t2grid` class added a `rectgeo()` method which creates a `mulgrid` geometry from a rectangular TOUGH2 grid. Grids must contain block centre co-ordinates and have a complete bottom layer. However, grids that have been translated or rotated, have incomplete upper layers (representing topography) or have inactive boundary condition blocks on the top or sides are supported.

In general, TOUGH2 grids do not necessarily follow the block naming conventions supported by the MULgraph geometry format. However, now such grids can still make use of PyTOUGH's grid geometry capabilities by means of block name mappings. The `rectgeo()` method produces a block name mapping which can be passed into various PyTOUGH functions (e.g. for post-processing), allowing other naming conventions to be supported.

2.6 New post-processing features

From its initial release PyTOUGH included functionality for producing 2-D plots of model grids, rock type information and model results, via the Python library Matplotlib. These plots over horizontal model layers and vertical slices can be exported to a variety of raster and vector image formats for production-quality output. The ability to script the creation of plots can greatly streamline the reporting process, particularly if the same set of plots has to be produced multiple times with updated models or datasets.

This 2-D post-processing capability has been enhanced by adding the ability to plot well tracks and flow arrows on plots. Flow arrows may represent either the raw fluxes through block faces output from TOUGH2, or block-centred average fluxes computed by assuming approximately tri-linear variation in flux variables over each block and performing least-squares fitting to the face fluxes, similar to the process described by Painter et al. (2012). This technique works for both structured and unstructured grids. An example of a PyTOUGH layer plot with block-averaged flow vectors is shown in Fig. 1.

In addition, it is now possible to display colour-coded rock types on 2-D PyTOUGH plots. In some applications, models contain large numbers of rock types, but many are variants of a smaller subset of basic types (e.g. based on geological rock units). PyTOUGH plots offer the option of grouping related rock types together for increased clarity. An example of a PyTOUGH slice plot with well tracks and grouped rock types is shown in Fig. 2.

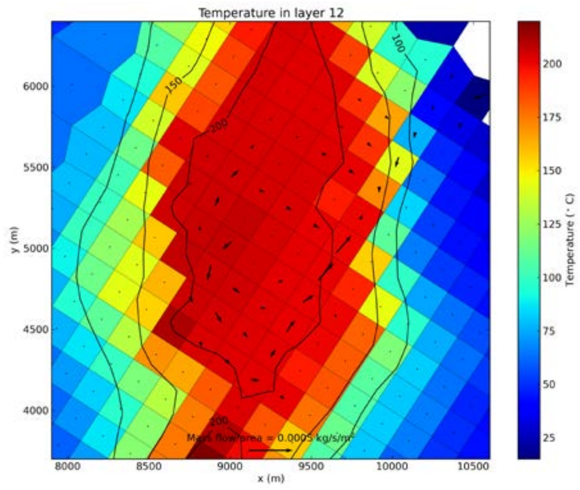

Figure 1: Example PyTOUGH layer plot output, showing temperature shading and contours, and block-averaged mass flux vectors

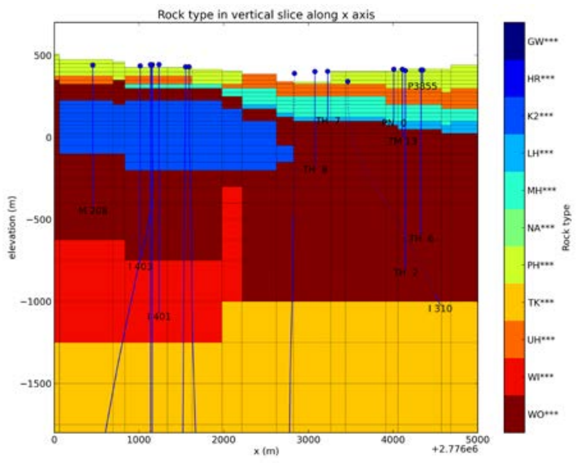

Figure 2: Example PyTOUGH vertical slice plot output, showing well tracks and grouped rock types

3. OTHER ENHANCEMENTS

Some of the other enhancements made to various aspects of PyTOUGH since its initial release are briefly listed below.

3.1 Grid geometry

- **Tilted grids**: PyTOUGH's `mulgrid` class now supports the tilting parameters in the MULgraph geometry format, used to create non-horizontal layered TOUGH2 grids.

- **Block naming**: more flexible block naming options are now included. Users can specify the character set used to create block names, allowing mixtures of upper-case, lower-case and non-alphabetic characters, which can be necessary for creating unique block names in very large models.

- **Layer refinement**: a simple function has been added for refining selected grid layers by a specified factor.

- **Grid reduction**: the `mulgrid.reduce()` method makes it easy to extract part of a model grid, corresponding to a set of specified columns.

- **Compound columns**: support for columns with more than four sides has been added to the `mulgrid` methods for fitting surface elevation and other scattered data to the grid columns, as well as exporting grid geometry, rock data and model output to VTK files suitable for 3-D visualization with Paraview or similar software.

- **Grid export**: `mulgrid` geometries may now be exported to ExodusII, a commonly used mesh and data-storage format, for interaction with other software.

3.2 TOUGH2 grids

- **Auxiliary mesh files**: TOUGH2 grids can now be read in from the auxiliary MESH file (or the TOUGH2-MP binary MESHA and MESHB files) instead of the main input data file.

- **Grid embedding**: a `t2grid.embed()` method class has been added for embedding one TOUGH2 grid inside another. This can be used, for example, to embed a radial model around a well within one block of a larger reservoir model.

- **Block re-ordering**: a `demote_block()` method has been added for moving a block to the end of the block list, mainly to make it easier to apply boundary conditions via inactive blocks.

3.3 TOUGH2 data files

- **Conversions between TOUGH2 and AUTOUGH2**: automatic conversions in both directions are now possible, handling not only the slightly different formats (e.g. additional sections) but also conversion of 'MOP' and other parameters.

- **Extra precision input**: AUTOUGH2 now has an option for reading input from auxiliary 'extra precision' files. Higher-precision floating point input can be useful in some situations, e.g. for inverse modelling when small changes in input parameters need to be resolvable. PyTOUGH's `t2data` class now supports reading and writing these extra precision files.

3.4 TOUGH2 output

- **Table skipping**: support has been added for skipping the parsing of specified tables from TOUGH2 output. This can speed up post-processing if e.g. the large connection tables are not needed for a particular post-processing task.

- **Output formats**: support has also been added for more of the many subtly different varieties of standard TOUGH2 output. There are small (but significant) format variations between output from different EOS modules, and also between TOUGH2 and iTOUGH2.

3.5 Speed improvements

- **Grid searching**: `mulgrid` methods for determining the column a 2-D point lies inside, and the node nearest to a 2-D point, have been made much more efficient by using optional quadtree and k-d tree data structures respectively.

- **Reading TOUGH2 data files**: low-level changes to the `t2data` class have resulted in some speed increases in reading in TOUGH2 input data files.

- **Slice plots**: the `mulgrid.slice_plot()` method now uses a form of particle-tracking algorithm to determine the trajectory of the slice across the grid columns.

3.6 Bug fixes

Fixes have been made for approximately 100 minor bugs since PyTOUGH's v.1.0 release. These and all other changes to the code may be viewed via the logs of the Git version control system on the PyTOUGH website (https://github.com/acroucher/PyTOUGH).

3.6 Installation

Installation of the PyTOUGH library has been simplified by the addition of a setup script. Regardless of computing platform (Linux, MS Windows, Mac OS etc.), running this Python script will install the software, with no further actions (e.g. setting environment variables) needed.

3.7 Documentation

The PyTOUGH user guide PDF has been substantially revised. Hyperlinks to referenced sections and external URLs have been added throughout. The text has been made more easily searchable, code examples are now syntax-highlighted, and there is a comprehensive index.

The wiki section on the PyTOUGH website has been expanded with lists of relevant conference and journal papers, tutorial material and example scripts.

3.8 Licensing

The PyTOUGH license has been changed from the GNU GPL to LGPL. This change allows commercial closed-source software to link to PyTOUGH more easily.

4. APPLICATIONS

4.1 Modelling applications

PyTOUGH has been used to assist with a range of published reservoir modelling studies, in locations as diverse as Lihir Island (O'Sullivan et al., 2011), Ohaaki (Clearwater et al., 2012), Mita in Guatemala (Feather and Malate, 2013), Rotorua (Ratouis et al., 2014), Waiotapu (Kaya et al., 2014a), Taupo-Reporoa Basin (Kaya et al., 2014b) and Habanero (Ayling et al., 2015).

O'Sullivan et al. (2013) described a number of other applications of PyTOUGH. In addition, Wellmann et al. (2014) coupled PyTOUGH with the inverse modelling software iTOUGH2 to perform uncertainty analysis on structural geological data.

4.2 Software applications

PyTOUGH is used to implement some of the TOUGH2 capability in the Leapfrog Geothermal software (http://www.leapfrog3d.com).

TIM (Yeh et al., 2013), a new graphical interface for TOUGH2, makes extensive use of PyTOUGH.

ACKNOWLEDGEMENTS

Some of the development of PyTOUGH was carried out to assist with work funded by Contact Energy.

REFERENCES

Ayling, B.F., Hogarth, R.A and Rose, P.E.:. Tracer testing at the Habanero EGS site, central Australia. *Geothermics*, in press (2015).

Barker, J.: A generalized radial flow model for hydraulic tests in fractured rock. *Water Resources Research* 24(10), 1796–1804 (1988).

Clearwater, E.K., O'Sullivan, M.J., Brockbank, K. and Mannington, W.I.: Modelling the Ohaaki geothermal system. *Proc. TOUGH Symposium 2012*, Berkeley, California, September 17-19 (2012).

Croucher, A.E. : PyTOUGH: a Python scripting library for automating TOUGH2 simulations. *Proc. 33rd NZ Geothermal Workshop*, Auckland, NZ (2011).

Feather, B.M. and Malate, R.C.M. (2013). Numerical modelling of the Mita geothermal field, Cerro Blanco, Guatematala. *Proc. 38th Workshop on Geothermal Reservoir Engineering*, Stanford University, Stanford, California, February 11-13 (2013).

Kaya, E., O'Sullivan, M.J. and Hochstein, M.P.:. A three dimensional numerical model of the Waiotapu, Waikite and Reporoa geothermal areas, New Zealand. *J. Volcanology Geoth. Res.* 283, 127-142 (2014a).

Kaya, E., O'Sullivan, M.J. and Yeh, A.: Three dimensional model of the deep geothermal resources in the Taupo-Reporoa Basin, New Zealand. *J. Volcanology Geoth. Res.* 284, 46-60 (2014b).

Moridis, G., Kowalsky, M. and Pruess, K.: TOUGH+HYDRATE v1.0 user's manual. Lawrence Berkeley National Laboratory report LBNL-161E (2008).

O'Sullivan, J., Croucher, A.E., O'Sullivan, M.J., Stevens, L. and Esberto, M.: Modelling the evolution of a mine pit in a geothermal field at Lihir Island, Papua New Guinea. *Proc. 33rd NZ Geothermal Workshop*, Auckland, NZ (2011).

O'Sullivan, J., Dempsey, D., Croucher, A.E., Yeh, A. and O'Sullivan, M.J.: Controlling complex geothermal simulations using PyTOUGH. *Proc. 38th Workshop on Geothermal Reservoir Engineering*, Stanford University, Stanford, California, February 11-13 (2013).

O'Sullivan, M. and Bullivant, D.: A graphical interface for the TOUGH2 family of flow simulators. *Proc. TOUGH Workshop 1995*, Lawrence Berkeley National Laboratory, University of California, Berkeley (1995).

Painter, S.L, Gable, C.W. and Kelkar, S.: Pathline tracing on fully unstructured control-volume grids. *Comput. Geosci.* 16, 1125-1134 (2012).

Pruess, K. : The TOUGH codes- a family of simulation tools for multiphase flow and transport processes in permeable media. *Vadose Zone Journal* 3(3), 738-746 (2004).

Pruess, K. and Narashimhan, T.N.: A practical method for modeling fluid and heat flow in fractured porous media. *Soc. Pet. Eng. J.* 25(1), 14-26 (1985).

Ratouis, T.M.P., O'Sullivan, M.J. and O'Sullivan, J.: An Updated Numerical Model of Rotorua Geothermal Field. *Proc. 39th Workshop on Geothermal Reservoir Engineering*, Stanford University, Stanford, California, February 24-26 (2014).

Wellmann, J.F., Croucher, A.E. and Regenauer-Lieb, K.: Python scripting libraries for subsurface fluid and heat flow simulations with TOUGH2 and SHEMAT. *Computers & Geosciences* 43, 197-206 (2012).

Wellmann, J.F., Finsterle, S. and Croucher, A.E.: Integrating structural geological data into the inverse modelling framework of iTOUGH2. *Computers & Geosciences*, 65, 95-109 (2014).

Yeh, A., Croucher, A.E. and O'Sullivan, M.J.:. TIM – yet another graphical tool for TOUGH2. *Proc. 35th NZ Geothermal Workshop*, Auckland, NZ (2013).

Xu, T., Spycher, N., Sonnenthal, E., Zhang, G., Zheng, L. and Pruess, K.: TOUGHREACT Version 2.0: A simulator for subsurface reactive transport under non-isothermal multiphase flow conditions. *Computers and Geosciences* 37(6), 763-774 (2011).