TIM – YET ANOTHER GRAPHICAL TOOL FOR TOUGH2

Angus Yeh, Adrian E. Croucher and Michael J. O'Sullivan

Department of Engineering Science, University of Auckland, Private Bag 92019, Auckland 1142, New Zealand

a.yeh@auckland.ac.nz

Keywords: *TOUGH2, MULgraph, reservoir modelling, GUI, visualisation, graphics, PyTOUGH, Python, PyQt*

ABSTRACT

A new open source graphical tool, called TIM, has been developed to aid the workflow of TOUGH2 reservoir model development and calibration. Visualisation of model parameters and simulation results is TIM's main feature. It also allows the user to manipulate the model data file interactively. In contrast to some other software tools which excel at qualitative 3-D visualisation and presentation, TIM mainly aims at producing easily accessible 2-D layer and 2-D slice plots that integrate the display of colour, text and flow arrows. This allows clearer quantitative assessment of local model behaviour which is often important for manual model calibration. In this respect, the software fills a gap among the available graphical pre-/post-processors for TOUGH2.

Instead of a more generic design focused on complete flexibility, the user interface is tailored to match the common workflow of model calibration. The operation of the software is designed to ease the cycle of parameter adjustment, simulation, and visualisation and plotting of results.

The software is written in the Python programming language, and makes use of the PyTOUGH library and the application framework PyQt. This allowed very rapid development, and provides cross-platform functionality and easy extensibility. The software has already proven very useful with only limited time and energy invested in its development.

1. INTRODUCTION

The TOUGH2 family of flow simulators has attracted the development of many pre- and post-processing graphical tools. MULgraph (O'Sullivan and Bullivant 1995) is one of the earliest graphical user interfaces (GUIs) designed for TOUGH2 model calibration. PetraSim (Swenson, Hardeman *et al*. 2002) provides a fully integrated environment in which modellers can pre-process, run, and post-process TOUGH2 models. GeoCad (Burnell, White *et al*. 2003) has functionality similar to that of MULgraph, plus some grid generation capabilities.

In recent years, more GUIs for TOUGH2 have been developed and published, possibly resulting from advancements in software technology: These include the Simple Geothermal Modeling Environment software of Tanaka and Itoi (2010), Leapfrog Geothermal (Alcaraz, Lane *et al*. 2011), Tougher (Li, Niewiadomski *et al*. 2011), mView (Avis, Calder *et al*. 2012), TOUGHVISUAL (Yang, Xu *et al*. 2012), and TOUGH2Viewer (Bondua, Berry *et al*. 2012). Most of these more modern GUIs provide three-dimensional (3-D) viewing. It is worth noting that mView is quite different from PetraSim in that, rather than creating a monolithic modelling environment, it emphasises a modular toolkit concept. The latter, modular approach is also the one adopted for our new interface.

We recognise that different visualisation styles are useful in different situations. Many of the more modern GUIs concentrate primarily on 3-D visualisation, which can be very useful for qualitative assessment and presentation. Our experience, however, is that two-dimensional (2-D) visualisation still offers benefits over 3-D visualisation in day to day model development. This is because 2-D layers and slices allow a clearer overlay of multiple model properties and results, including scalar and vector quantities, using colours, text and flow arrows on the model grid. This allows a more detailed quantitative assessment, which is often important in model calibration. Additionally, locating and isolating a specific area of interest in a full 3-D visualisation is often difficult. For this reason, MULgraph has continued to be an important post-processing tool in our manual calibration workflow.

Originally developed in 1995, MULgraph is showing its limitations in the current modelling environment. One major limitation of MULgraph is its linear style of operation, with users having to carry out operations in a predetermined sequence. For example, if temperatures are displayed in a certain model layer, the user is required to exit and then re-enter through several levels of menus in order to display temperatures in a different layer. A similarly lengthy procedure is required to switch from displaying variables defined at blocks (e.g. pressures and temperatures) to variables defined at connections (e.g. mass flows), or to time histories of variables at a particular block or generator. It is not possible to see several different kinds of information displayed at the same time, or even switch between them quickly. This sequential mode of operation is unfortunately built into the core architecture of the MULgraph code, and would be very difficult to modify. At present MULgraph users have to print out separate plots of, say, temperature, permeability and mass flows and look at three paper plots in order to decide how to change parameters to make part of a model hotter or colder.

MULgraph was written in Fortran77, and hence in a non-object-oriented programming style. This makes it difficult to maintain and extend its capabilities. The use of the Graphical Kernel System (GKS) and other low level graphics libraries in MULgraph also makes porting the code to modern operating systems difficult. For the same reason, the dated look and feel of the interface is almost impossible to update without major work. It was thought that overall, the effort of improving MULgraph in any significant way was probably comparable to, or greater than, that of developing replacement software and this conclusion provided the motivation for starting the software project described here.

In recent years, PyTOUGH (Croucher 2011; Wellmann, Croucher *et al*., 2012), a Python scripting library for TOUGH2, has proven to be very useful for manipulating TOUGH2 simulations (O'Sullivan *et al*., 2013). The Leapfrog Geothermal GUI (Alcaraz, Lane *et al*., 2011) already uses PyTOUGH to load simulation results for visualisation. Our experiences in using PyTOUGH's extensive abilities for handling TOUGH2 input and output files soon inspired the idea of building a new GUI, that replicates and extends the capabilities of MULgraph using PyTOUGH as its foundation.

The longevity of MULgraph, despite its limitations, suggests that a modern replacement will fill a gap among the available graphical pre- and post-processors for TOUGH2. A prototype was built to investigate the feasibility of the idea. This has quickly proved to be useful, even with only limited time and energy invested. In this paper, we describe the design, implementation and features of the software.

2. SOFTWARE DESIGN

The new software, referred to here as TIM (TIM Isn't MULgraph), is designed around a few key requirements:

2.1 Optimised for model calibration workflow

As mentioned above, TIM retains the 2-D visualisation approach of MULgraph. In MULgraph's case, the 2-D approach probably resulted in part from the limitations of the graphical software available at the time of its development, but in fact resulted in a very effective tool for quantitative model assessment and manual model calibration.

TIM's main features are loosely based on those of MULgraph. It is, however, built using a very different approach, and abandons MULgraph's sequential operation. Through the use of text, colour and flow arrows, up to two scalar quantities and one vector quantity can be displayed simultaneously and all different model parameters and simulation outputs can be accessed. Information integration and clarity of presentation have the highest priority.

The main view allowed by TIM consists of 'scenes' that display one or more views of the model and results. It is quick and easy to switch between scenes or even to show multiple scenes together. This main view can be accompanied by various utility windows, such as line plots or tables, in a synchronous manner. The facilitation of better calibration workflow is achieved by allowing tasks to be accomplished in any desired order. In short, TIM's goal is to allow modellers access to all model information, in any order, with the only limitation being the 2-D visualisation style.

Separately, the occasional need for 3-D visualisation has been achieved using Leapfrog Geothermal or ParaView (Ahrens, Geveci *et al.* 2005; ParaView 2013) (with PyTOUGH as a file converter).

2.2 Rapid software development

The amount of time and human resource required for TIM's past and future development was and is obviously an important consideration. Unlike software written in the technology environment of 1980s, modern programs are often built on software libraries. This approach avoids programmers having to "reinvent the wheel", and allows them to make use of libraries they may have already used in a different context, and hence are already familiar with. Today, there are plenty of high quality open source libraries available. In particular, this is true for all the key components of TIM: the GUI, visualisation engine, TOUGH2 interface, and scientific plotting.

2.3 Easy extensibility

Making the software easily extensible, both by its developers and by the user, means the application can be constantly optimised for workflow, even when the workflow changes over time, or between different modellers. The choice of well established, easy to use, software libraries has ensured the extensibility of the software.

2.4 User-friendly modern interface

MULgraph's idiosyncratic user interface is, for the user, one of its most obvious limitations. In its place, we wanted an interface that was modern and would feel familiar to its users. The common operations that work well in other visualisation software, such as view control and selection, are also used in TIM, so it is more natural for users to switch between applications. However, TIM's user interface concentrates on modelling by using the terminology and operational logic familiar to a modeller.

The user-friendliness of an application is sometimes hard to measure, especially when considering the different requirements of new users and experienced modellers who wish to optimise workflow. It is important to improve the user-friendliness constantly by iteratively performing usability testing and modifying or redesigning the interface. We have achieved this aim by involving several colleagues as beta-testers. We have made various changes to TIM based on problems our beta-testers have experienced. This kind of iterative development is only possible because of the rapid development platform we are using.

3. IMPLEMENTATION

TIM is written in Python, a versatile, powerful, modern programming language that is open source and easy to learn. Python programs, unlike those written in Fortran or C/C++, do not require compilation, which speeds the development cycle and makes it easier for users to customise and extend the software to their particular and changing needs. Using Python gives TIM a different level of extensibility compared with other GUIs for TOUGH2.

Another attractive feature of Python is the very large number of software libraries available for it, most of them open source, extending the capabilities of the language into very diverse areas. The implementation of TIM is based on the combination of two of these Python libraries: PyQt and PyTOUGH.

The GUI part of TIM makes use of PyQt, a Python binding of the GUI framework Qt. Qt is a popular cross-platform application framework and GUI toolkit. Python binding enables code using the powerful Qt C++ library to be more easily written and debugged (Summerfield 2008). The use of PyQt is one of the key choices that enabled TIM to be very rapidly developed.

TIM uses PyTOUGH to interface with the input and output of TOUGH2 simulations. PyTOUGH is actively maintained and continues to support a variety of TOUGH2 simulators. Hence, TIM is also able to support them, and automatically benefits from the continual improvements made to PyTOUGH. If needed, PyTOUGH itself can also be modified by the user, because it is an open source library.

Using Python and PyTOUGH also naturally opens up the possibility of powerful scripting capabilities within TIM. TIM includes Python's interactive console to allow users to interact dynamically with objects of both the GUI components and PyTOUGH's objects. This should be a natural function for modellers who already use PyTOUGH.

Since Python, PyQt and PyTOUGH all have cross-platform capabilities, TIM inherits their advantage of being highly portable. No change in code is required between computing platforms, including the three major environments that TOUGH2 is commonly run on: Windows, Linux, and Mac-OS.

4. GEOMETRY

In the finite volume approach used by TOUGH2, the model grid is defined by a set of block volumes and the connections between them, without any reference to a coordinate system. Except in the case of very simple regular rectangular grids, a separate geometry definition is usually needed (or at least very useful) for creating the TOUGH2 grid and visualising the results. At present, TIM uses the same 'geometry file' as MULgraph for this purpose. This assumes a layer/ column grid structure, with a number of horizontal layers, each of constant thickness and with the same horizontal structure in plan-view. Arbitrary unstructured horizontal grid structures are permitted. It is also possible to have incomplete layers at the top of the model, to represent either varying topography or the elevation of the water table.

With a layered grid structure of this kind, it is natural for modellers to want to view the model in terms of layers, i.e. one layer at a time. Accordingly TIM, like MULgraph, offers scenes based on layers, and also scenes representing vertical slices through the model grid. TIM additionally allows the user to create vertical slices based on polylines (rather than just simple straight lines). Multiple slice scenes can be created, and the user can easily switch between them (as well as the layer scenes).

Although TIM currently creates its visualisation scenes using a MULgraph geometry file, the core code architecture is not dependent on this particular geometry definition. The visualisation engine simply creates scenes by working out which TOUGH2 blocks and connections should be in the scene, and how they should be presented. The scenes can then show model data (i.e. the simulation input and output) by mapping them to these blocks and connections. Hence, it would be easy to extend TIM to support other types of grid geometry, as long as 2-D scenes were still meaningful in terms of model calibration, or to other file formats for layer/ column geometries. This could be done by either extending the PyTOUGH library, or simply writing code to convert other formats into MULgraph geometry files. In fact, PyTOUGH already supports conversions from the 2-D Gmsh grid format (Geuzaine and Remacle 2009) to a MULgraph geometry file.

5. FEATURES

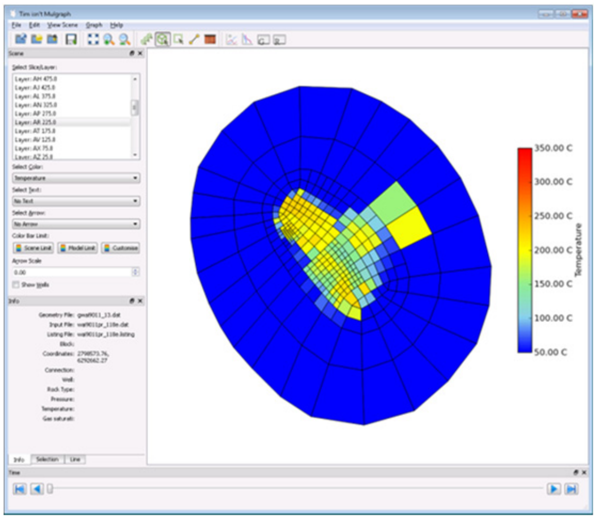


Figure 1: TIM's main window can display any scene and is highly customisable.

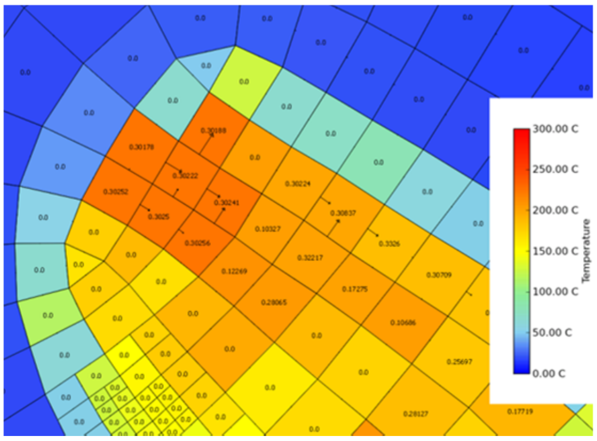


Figure 2: A layer scene shows temperature in colour, gas saturation in text, and mass flow using arrows.

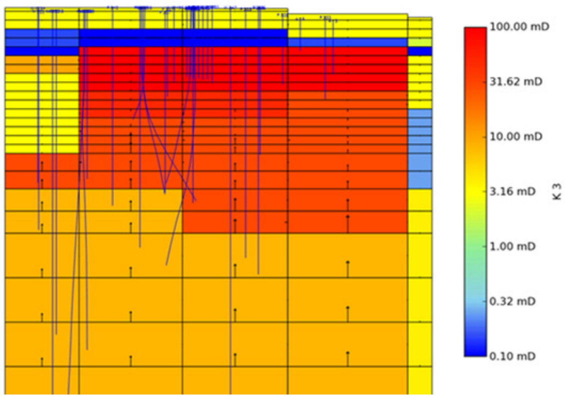
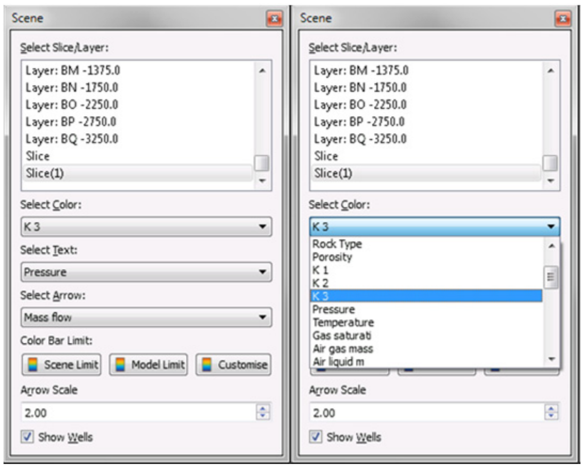


Figure 3: A vertical slice scene shows well tracks, vertical permeability, and mass flow.


Scene

Select Slice/Layer:

Layer: BM -1375.0
Layer: BN -1750.0
Layer: BO -2250.0
Layer: BP -2750.0
Layer: BQ -3250.0
Slice
Slice(1)

Select Color: K 3

Select Text: Pressure

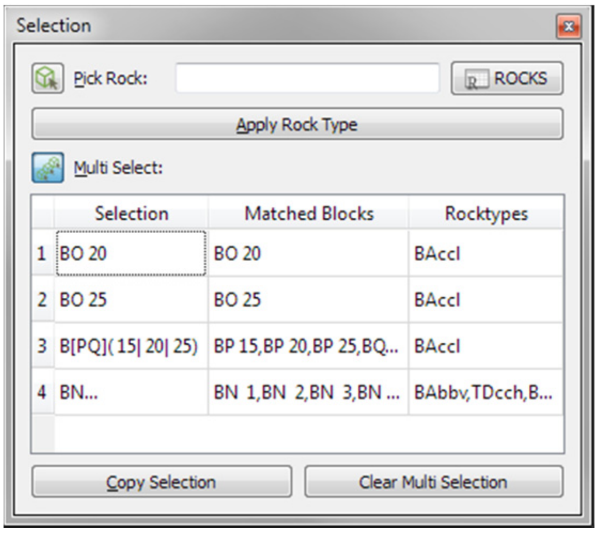
Select Arrow: Mass flow

Color Bar Limit: Scene Limit | Model Limit | Customise

Arrow Scale: 2.00

☑ Show Wells

Figure 4: The scene control panel allows user to independently switch between scenes, variables for colour, text, and arrow. The list of displayable variables is user customisable.


Selection

Pick Rock: | ROCKS

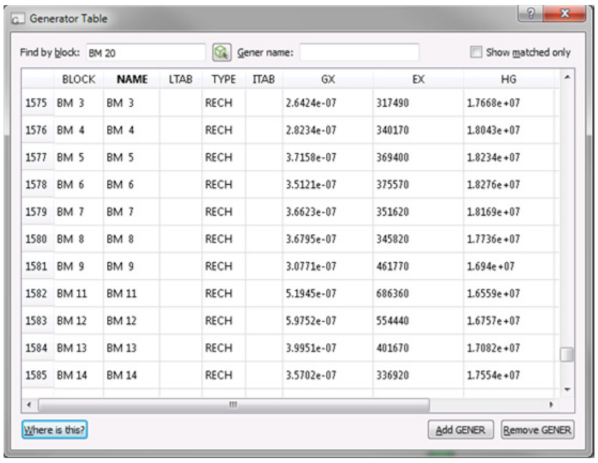
Apply Rock Type

Multi Select:

	Selection	Matched Blocks	Rocktypes
1	BO 20	BO 20	BAccl
2	BO 25	BO 25	BAccl
3	B[PQ](15| 20| 25)	BP 15,BP 20,BP 25,BQ...	BAccl
4	BN...	BN 1,BN 2,BN 3,BN ...	BAbbv,TDcch,B...

Copy Selection | Clear Multi Selection

Figure 5: Selection can be done in various ways, including intuitive methods like clicking on blocks, and more flexible methods like using regular expressions (a string pattern matching language).


Generator Table

Find by block: BM 20 | Gener name: | ☐ Show matched only

	BLOCK	NAME	LTAB	TYPE	ITAB	GX	EX	HG
1575	BM 3	BM 3		RECH		2.6424e-07	317490	1.7668e+07
1576	BM 4	BM 4		RECH		2.8234e-07	340170	1.8043e+07
1577	BM 5	BM 5		RECH		3.7158e-07	369400	1.8234e+07
1578	BM 6	BM 6		RECH		3.5121e-07	375570	1.8276e+07
1579	BM 7	BM 7		RECH		3.6623e-07	351620	1.8169e+07
1580	BM 8	BM 8		RECH		3.6795e-07	345820	1.7736e+07
1581	BM 9	BM 9		RECH		3.0771e-07	461770	1.694e+07
1582	BM 11	BM 11		RECH		5.1945e-07	686360	1.6559e+07
1583	BM 12	BM 12		RECH		5.9752e-07	554440	1.6757e+07
1584	BM 13	BM 13		RECH		3.9951e-07	401670	1.7082e+07
1585	BM 14	BM 14		RECH		3.5702e-07	336920	1.7554e+07

Where is this? | Add GENER | Remove GENER

Figure 6: TOUGH2 input files can be edited using the tables.

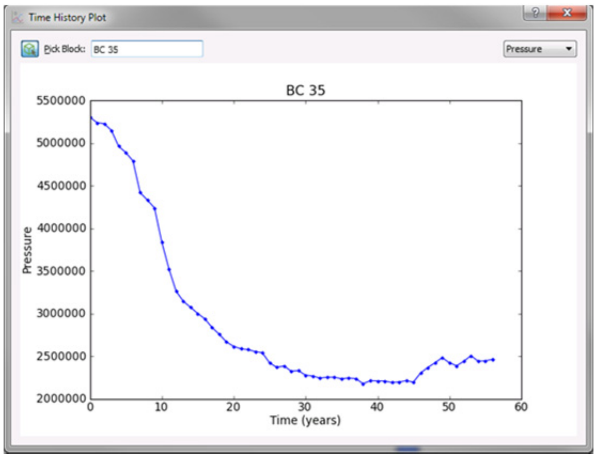
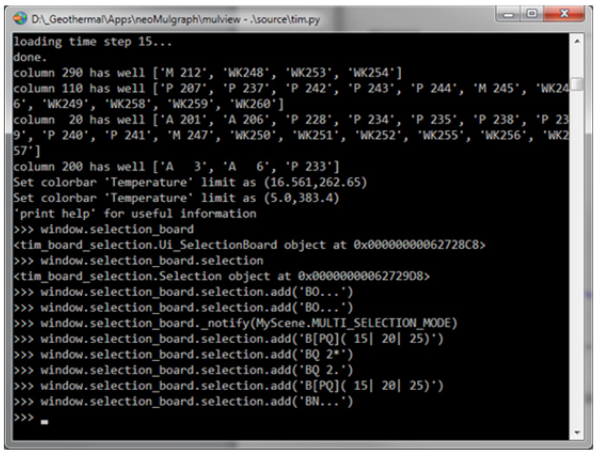


Figure 7: Graphs such as history plots and well downhole plots can be independently shown alongside the current scene.


D:_Geothermal\Apps\neoMulgraph\mulview - .\source\tim.py

```
loading time step 15...
done.
column 290 has well ['M 212', 'WK248', 'WK253', 'WK254']
column 110 has well ['P 207', 'P 237', 'P 242', 'P 243', 'P 244', 'M 245', 'WK24
6', 'WK249', 'WK258', 'WK259', 'WK260']
column  20 has well ['A 201', 'A 206', 'P 228', 'P 234', 'P 235', 'P 238', 'P 23
9', 'P 240', 'P 241', 'M 247', 'WK250', 'WK251', 'WK252', 'WK255', 'WK256', 'WK2
57']
column 200 has well ['A   3', 'A   6', 'P 233']
Set colorbar 'Temperature' limit as (16.561,262.65)
Set colorbar 'Temperature' limit as (5.0,383.4)
'print help' for useful information
>>> window.selection_board
<tim_board_selection.Ui_SelectionBoard object at 0x00000000062728C8>
>>> window.selection_board.selection
<tim_board_selection.Selection object at 0x00000000062729D8>
>>> window.selection_board.selection.add('BO...')
>>> window.selection_board.selection.add('BO...')
>>> window.selection_board._notify(MyScene.MULTI_SELECTION_MODE)
>>> window.selection_board.selection.add('B[PQ]( 15| 20| 25)')
>>> window.selection_board.selection.add('BQ 2*')
>>> window.selection_board.selection.add('BQ 2.')
>>> window.selection_board.selection.add('B[PQ]( 15| 20| 25)')
>>> window.selection_board.selection.add('BN...')
>>> 
```

Figure 8: TIM's interactive mode is available for users to interact with PyTOUGH objects (or GUI components) directly using the Python language.

6. ACCESS

The TIM software is currently in an initial development and testing stage, and the authors intend to release it in early 2014. The code will be open source, released under the GNU General Public License (GPL). Users will be encouraged to customise the software for their workflow or extend its capability.

7. CONCLUSIONS

A new open source GUI, called TIM, has been developed to aid the workflow of TOUGH2 reservoir model development and calibration. TIM is designed differently from many available GUIs for TOUGH2. With a clear 2-D visualisation style, it enables modellers to quantitatively assess multiple model properties and results, including scalar and vector quantities, using colours, text and flow arrows on the model grid. Together with functions specifically designed by and for modellers, this makes TIM very well suited to the workflow of model development and calibration.

TIM is written in the Python scripting language and makes use of the PyTOUGH library. This gives TIM a very rapid development cycle and a higher level of extensibility and customisability than other GUIs for TOUGH2.

REFERENCES

Ahrens, J., Geveci, B., Law, C. *ParaView: An End-User Tool for Large Data Visualization*. In the Visualization Handbook. Edited by C.D. Hansen and C.R. Johnson. Elsevier. (2005).

Alcaraz, S., Lane, R., Spragg, K. et al.: *3D geological modelling using new LEAPFROG geothermal software*. Thirty-Sixth Workshop on Geothermal Reservoir Engineering, Stanford University, Stanford, California. (2011).

Avis, J., Calder, N. and Walsh, R.: *mView - a powerful pre- and post-processor for TOUGH2*. TOUGH Symposium 2012, LBNL, Berkeley, California. (2012).

Bondua, S., Berry, P., Bortolotti, V. and Cormio, C. *TOUGH2Viewer: a post-processing tool for interactive 3D visualization of locally refined unstructured grids for TOUGH2*. Computers & Geosciences **46**, 107-118 (2012).

Burnell, J. G., White, S. P., Osato, K. and Sato, T.: *Geo-Cad, a pre- and postprocessor for TOUGH2*. TOUGH Symposium 2003, LBNL, Berkeley, California. (2003).

Croucher, A.: *PyTOUGH: a Python scripting library for automatic TOUGH2 simulations*. Proceeding 33rd New Zealand Geothermal Workshop 2011, Auckland, New Zealand. (2011).

Geuzaine, C., Remacle, J.-F. *Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities*. International Journal for Numerical Methods in Engineering **79**(11), 1309-1331 (2009).

Li, Y., Niewiadomski, M., Trujillo, E. and Sunkavalli, S. P. *Tougher: a user-friendly graphical interface for TOUGHREACT*. Computers & Geosciences **37**, 775-782. (2011).

O'Sullivan, M. J., Bullivant, D. P.: *A graphical interface for the TOUGH family of flow simulators*. Proceedings of the TOUGH Workshop '95, Berkeley,California. (1995).

O'Sullivan, J.P., Dempsey, D., Croucher, A.E., Yeh, A. and O'Sullivan, M.J. *Controlling complex geothermal simulations using PyTOUGH.* Proceedings of the Thirty-Eighth Workshop on Geothermal Reservoir Engineering, Stanford University, Stanford, California. (2013).

ParaView, http://www.paraview.org Cited August. (2013).

Summerfield, M. Rapid GUI programming with Python and Qt : the definitive guide to PyQt programming. Upper Saddle River, NJ, Prentice Hall. (2008).

Swenson, D., Hardeman, B., Butler, S., Persson, C. and Thornton, C.: *Using the Petrasim pre- and post-processing for TOUGH2, TETRAD, and STAR*. Twenty-Seventh Workshop on Geothermal Reservoir Engineering, Stanford University, Stanford, California. (2002).

Tanaka, T., Itoi, R.: Development of numerical modeling environment for TOUGH2 simulator on the basis of graphical user interface (GUI). Proceedings World Geothermal Congress 2010, Bali, Indonesia. (2010).

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

Yang, Y., Xu, T., Wang, F. et al.: TOUGHVISUAL: *a user-friendly pre-processing and post-processing graphical interface for TOUGHREACT*. TOUGH Symposium 2012, LBNL, Berkeley, California. (2012).