GEOTHERMAL SUPERMODELS PROJECT: AN UPDATE ON FLOW SIMULATOR DEVELOPMENT

Adrian Croucher[1], Michael J. O'Sullivan[1], John O'Sullivan[1], Justin Pogacnik[1], Angus Yeh[1], John Burnell[2] and Warwick Kissling[2]

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

[2]GNS Science, Private Bag 30-368, Lower Hutt 5040, New Zealand

a.croucher@auckland.ac.nz

Keywords: *Reservoir models, numerical modelling, flow simulator*

ABSTRACT

The "Geothermal Supermodels" project is a four-year New Zealand-based research programme focusing on developing next-generation integrated geothermal modelling tools. A major part of this programme is the development of a new open-source geothermal reservoir flow simulator, featuring modern modular object-oriented code design, parallelized assembly and solution of equations, leverage of established numerical libraries, improved input and output and improved convergence of natural state models.

This paper describes the software design of this new simulator, the software development workflow (which includes integrated unit testing and automated generation of documentation) and progress to date on implementation, together with preliminary output from a test problem.

1. INTRODUCTION

The "Geothermal Supermodels" project is a four-year research programme based in New Zealand, aiming to develop next-generation integrated geothermal modelling tools. These include a new flow simulator, geophysical and geochemical codes, together with the linkages between them (Burnell et al., 2015).

This paper focuses on the flow simulator component of the project, discussing currently-available simulators suitable for geothermal reservoir modelling, and outlining the specification for a new simulator. Details are given of the software design, development workflow and progress to date on implementation, as well as preliminary results from a demonstration problem.

2. FLOW SIMULATOR REQUIREMENTS

2.1 Thermodynamic and numerical requirements

Fluid flows in geothermal systems are difficult to model numerically, largely because of the highly non-linear processes taking place. Fluids may undergo large changes in temperature, often resulting in phase transitions and zones of multi-phase flow. Numerical simulators for modelling these systems must not only be capable of representing this complex thermodynamic behaviour, but must also be based on numerical methods robust enough to cope with the additional computational challenges posed by geothermal problems.

Of the few software packages available that meet these requirements, probably the best-known and most widely used for geothermal reservoir modelling is the TOUGH2 simulator (Pruess, 2004). It is capable of modelling multi-phase, multi-component flows at temperatures up to around 350°C, and its algorithms for handling phase changes are robust in most situations. It is based on an "integrated finite difference" (or finite volume) numerical method which, while being of relatively low order accuracy, is reliable and applicable to unstructured meshes, which are often useful for modelling e.g. complex domains or areas of local refinement.

The TOUGH2 code dates back to the 1980s, and its longevity and the shortage of competing codes in the geothermal modelling sector are indications of the level of difficulty involved in modelling geothermal systems, and of the robustness of the numerical formulation TOUGH2 uses.

However, in recent years the need to solve larger and more demanding problems has become increasingly common. As a result, new simulation software requirements have emerged and the limitations of older software like TOUGH2 have become more apparent. Some of these requirements are detailed below.

2.2 Additional requirements

2.2.1 Large models

Present-day geothermal reservoir models regularly call for numbers of grid blocks in the range 10^4 - 10^5, still small compared with models used for example by the petroleum industry, but probably not envisaged when software like TOUGH2 was written. Progressing to larger, more detailed models is not practical without code able to run in parallel on multi-processor desktop machines or larger compute clusters.

Parallel versions of TOUGH2 have been developed, e.g. TOUGH2-MP (Zhang et al, 2003) and TOUGH+ (Moridis et al., 2008), but in our view retro-fitting parallel capability to existing serial code is not only very difficult, but usually not as successful as code designed and written in parallel from the start. In addition, the multi-processor versions of TOUGH2 are still hampered by other design limitations of the original code (see below).

2.2.2 Complex production scenarios

Simulations of geothermal production scenarios requires the ability to model complex arrangements of wells (generators) whose flow rates are influenced by reservoir conditions, flow rates in other wells and management constraints. Makeup wells need to be added, as needed, into the model. None of this is possible with TOUGH2 in its standard form. Variants such as AUTOUGH2 (Yeh et al., 2012) do include some capability for it, but the convenience and flexibility of these extra options is again limited by the original code design.

2.2.3 Input and output

The input for TOUGH2 (even in its more recent incarnations) comes from a fixed-format text file. The fixed-width formats are relatively easily machine-readable (and writeable) but can easily lead to subtle errors when edited manually.

Output from TOUGH2 is also in the form of formatted text files. Particularly for large models, this make the output files large in size and slow to post-process. Furthermore, there is little standardization of the particular output formats between different versions of TOUGH2 or even between different equation-of-state modules, which makes machine reading for post-processing unnecessarily difficult.

More efficient and effective options for input and output formats, which avoid these problems, are now available and will be described below.

2.2.4 Thermodynamic formulation

TOUGH2 (and most other simulators used for non-isothermal subsurface modelling) use the IFC-67 thermodynamic formulation for calculating the properties of water. However, this has now been superseded by the IAPWS-97 formulation (Wagner et al., 2000), which is faster and more accurate. In some situations the differences in results between the two formulations can be significant. Ideally, it would be useful to have both available, with IAPWS-97 for general use and IFC-67 for backwards compatibility and benchmarking purposes.

2.2.5 Convergence of natural state models

One of the most problematic aspects of geothermal reservoir modelling is obtaining a stable natural state solution, usually by trying to run the natural state TOUGH2 model with adaptive time stepping until the time step size is large. In many cases a large time step size is, in practice, difficult to reach. Sometimes this may be a result of complex rock structures in the model or the movement of boiling zones or air/water interfaces through the mesh.

In other cases, however, it can arise from subtle inconsistencies in the way thermodynamic properties are calculated by TOUGH2 during phase transitions (O'Sullivan et al., 2013). Hence, there is room for improvement in the way these calculations are carried out. Some such improvements have recently been added to the AUTOUGH2 simulator.

Additionally, the solution of the linear equations (formed during each iteration of the solution of the governing non-linear equations at each time step) can fail, particularly as the model approaches a steady state and the linear equations become increasingly ill-conditioned. Improved linear equation solvers and preconditioners for difficult problems are an area of active research and some are available via commonly-used numerical libraries (see below). Use of such libraries, rather than using built-in solvers, would allow access to these new methods.

2.2.6 Extended modelling capability

For some problems, extended modelling capability is needed, beyond that provided by a flow simulator like TOUGH2 in its basic form. For example, some models may require the simulation of rock mechanics or the movement of chemical species. For deeper models, supercritical capability may be need to handle very high temperatures and pressures.

Ideally, it should be possible to add such extra capability to the simulator using a modular approach, as has been used in multi-physics codes such as MOOSE (Podgorney et al., 2010). Unfortunately the main TOUGH2 code was written in an era when this was not yet possible, at least within the limitations of the Fortran 77 language it was written in. More modern languages and coding paradigms make it easier to develop clear, modular code. However, this is not something that can easily be retro-fitted to existing software. Adding significant extra capability to TOUGH2 has generally involved 'forking' the entire code (i.e. creating a separate version with the extra capability added), e.g. chemical transport via TOUGHREACT (Xu et al., 2006) or supercritical capability via the supercritical version of AUTOUGH2 (Croucher and O'Sullivan, 2008).

2.2.7 Licensing

TOUGH2 (although not TOUGH+) is distributed with source code, which has enabled users to understand how the code works and have confidence in the algorithms used. It has also enabled users to modify the code as needed to solve particular problems.

However, there are no requirements in the TOUGH2 license for users to share modifications with the original developers. This has contributed to the fragmentation of the code into multiple independently forked versions.

Many newer scientific software projects use open-source “copyleft” licenses (e.g. GNU GPL or LGPL) which allow users to make modifications, but require these to be shared with the original developers, discouraging fragmentation. Using a standardized open-source license makes the code accessible to industry users, academic researchers and students alike, and makes the terms easier for users to interpret.

2.3 Other simulators

Other simulators have been more recently developed with capabilities approaching the requirements listed above for geothermal simulation. These include DuMu^x (Flemisch et al., 2007), OpenGeoSys (Kolditz et al., 2012), PFLOTRAN (Mills et al., 2007) and OOMPFS (Franz, 2015).

However, while these codes can simulate multi-phase flows, most do not have the simulation of high-temperature geothermal systems as an explicit goal, and hence do not appear to have the phase-changing capabilities needed. For the same reason, they generally do not have off-the-shelf ability for simulating complex production scenarios.

The exception is OOMPFS, which was developed specifically for geothermal applications, but unfortunately is at present neither parallelized nor open-source. Hence, the decision was taken to develop a new geothermal flow simulator.

3. SOFTWARE DESIGN

From the requirements laid out above, it was clear that the new flow simulator would need a modern modular, object-oriented code architecture. Parallel capability would also need to be built in to the basic design, rather than added later.

We also wanted to make use of existing software libraries for numerical computation. We chose to base our simulator on the PETSc library (Balay et al., 2015), a highly-regarded and long established open-source library for scientific computation, first released in 1995. The PETSc library is callable from the C/C++, Fortran and Python programming languages and includes parallelized vector and matrix data structures, together with parallel tools for linear and non-linear equation solution, data management on structured and unstructured distributed meshes, and more. PETSc is used by a number of well-known simulators including PFLOTRAN, which has demonstrated good scalability on very large simulation problems (up to 10^9 unknowns and 60,000 processors).

For the choice of programming language, we needed a language that would allow the kind of modular object-oriented code architecture described above, without compromising performance (i.e. computation speed). Many recent codes, e.g. DuMux, OpenGeoSys and OOMPFS use the C++ language. However, the flexibility of C++ means that special measures (e.g. extensive use of templates) need to be taken to achieve good performance.

The latest revisions of the Fortran language standard, Fortran 2003 and 2008, include substantial new support for object-oriented programming. This means it is now possible to take advantage of Fortran's high numerical efficiency in an object-oriented setting. Like the developers of PFLOTRAN, we chose to use Fortran 2003 as the language for our project.

4. SOFTWARE DEVELOPMENT

We have adopted a test-driven software development process, which integrates unit testing (i.e. testing of individual subroutines) into the code development cycle. This is done using the FRUIT library for Fortran unit testing (http://sourceforge.net/projects/fortranxunit/), supplemented by a Python interface called FRUITPy that we developed and released as a standalone open-source project (https://github.com/acroucher/FRUITPy).

We use GNU Make as the software build system, as it is easy to integrate this with PETSc, and the Git tool for software version control. Automatic software documentation (for developers, rather than users) is generated by a relatively new tool called FORD (https://github.com/cmacmackin/ford), which is aimed specifically at Fortran code documentation.

5. SOFTWARE IMPLEMENTATION

5.1 Numerical formulation

The code uses a finite volume numerical formulation similar to that used by TOUGH2, solving for cell-averaged primary thermodynamic variables and computing fluxes between cells using a simple two-point approximation and upstream weighting. However, the code is flexible enough to permit experimentation with other formulations.

Time stepping is handled in a modular fashion, abstracted out of the remainder of the code, to permit the use of different numerical time stepping schemes. Currently the standard backward Euler scheme (as used by TOUGH2) is implemented, as well as the BDF2 method, which is a variable-stepsize multistep method with stability properties and computational cost similar to backward Euler, but second-order accuracy. In future we plan to investigate the use of exponential Euler time stepping methods (Pope, 1963), as well as schemes for the direct solution of steady-state problems without time stepping.

5.2 Mesh handling and data structures

The PETSc library recently introduced support for unstructured meshes distributed over multiple processors, via a new DMPlex class. This is used for storing mesh data, handling parallel mesh partitioning, and creation of parallel vectors and matrices on the model mesh.

By default, our code will require a geometric mesh to be specified. This is in contrast with TOUGH2 which only requires the finite volume mesh data (volumes, connection areas and connection distances) to be input. However, in most cases an auxiliary geometric mesh is needed for preparation of the finite volume mesh data and for post-processing. In addition, a geometric mesh is needed for some types of extended modelling capability, e.g. rock mechanics.

Over the mesh, three main parallel vectors are defined for storing the solution (primary thermodynamic variables in each cell), fluid properties and rock properties respectively. Fluid properties in each cell are calculated by the equation of state from the primary variables before each evaluation of the mass and energy balance equations. Rock properties in each cell can vary with time and be independently specified, or initialized from rock types as in TOUGH2.

Evaluation of the mass and energy balance equations in each cell is carried out using object-oriented local-level data structures, which contain pointers into the global parallel data vectors. Indexing into these vectors is handled by the DMPlex class.

5.3 Thermodynamics and equations of state

An abstract thermodynamics class is defined and sub-classed to implement the specific IFC-67 and IAPWS-97 thermodynamic formulations. This ensures both formulations have a consistent interface, and aside from selecting the desired formulation (based on input data) no further code is needed to handle multiple formulations.

Similarly, an abstract equation of state (EOS) class is defined and sub-classed for specific equations of state. The aim of this approach is to ensure as much code re-use, and hence consistency, as possible between different EOS modules. To date only pure water EOS modules (isothermal and non-isothermal) have been implemented.

5.4 Phase transitions

As in TOUGH2, phase transitions are handled using variable switching, though at the time of writing this is not yet fully implemented. As noted above, if a high degree of consistency is not maintained when re-calculating fluid properties after the transition, the non-linear equation solver may fail to converge, and this is one of the factors impeding the progress of TOUGH2 steady state solutions.

In TOUGH2, the equation of state module also calculates fluid properties in phases which are not physically present, as a way of simplifying the calculation of fluxes between cells with different phase conditions. This is a reasonable approach for sub-critical fluid, but becomes cumbersome for transitions to and from supercritical fluid, in which there is only one possible phase. To address this, we are considering a modified flux calculation which would avoid the need for calculating fluid properties in absent phases.

5.5 Sources and sinks

Currently only simple constant-rate sources and sinks have been implemented. In future we plan to introduce something along the lines of a higher-level “source manager” object to handle the complex source and sink configurations needed for modelling production scenarios, in which generation rates may be influenced by reservoir conditions, other sources or sinks, or management constraints. This would avoid the need for the proliferation of generator types introduced into AUTOUGH2 for handling specific situations.

5.6 Boundary conditions

Flux (i.e. Neumann) boundary conditions are most easily modelled, as in TOUGH2, using sources and sinks. For Dirichlet boundary conditions (e.g. specifying pressure and temperature), TOUGH2 requires the use of extra cells that are either “inactive” or have large volume to prevent their properties from changing during the simulation. However, this can complicate pre- and post-processing, because the physical geometric mesh does not contain such cells.

The PETSc DMPlex class facilitates the use of “ghost cells” for the application of Dirichlet boundary conditions (and are also used for parallel communication between cells on different processors). As these are created internally, they need not be part of the model input. Instead, boundary conditions can be specified explicitly and independently of the mesh definition.

5.7 Solution of linear and non-linear equations

At each time step, regardless of the time-stepping method used, the updated primary thermodynamic variables at the end of the time step are computed by solving the non-linear mass and energy balance equations. We use the PETSc SNES (Scalable Non-linear Equation Solver) class to solve the equations in parallel. This can use a standard Newton-Raphson method, with or without line searching, or other techniques as desired.

At each iteration the Jacobian matrix (the derivatives of the mass and energy balance equations with respect to the primary variables) must be evaluated. The PETSc SNES can optionally calculate the Jacobian itself using finite differencing, and at present we are using this option. In the longer term we plan to investigate computing the Jacobian analytically, as has been done in some other codes such as FEHM (Zyvoloski, 2007), to improve convergence of the non-linear equation solution.

Finally, at each iteration the solution is updated by solving a set of linear equations. For typical problems this is where a simulator will spend most of its computational time. By default the SNES uses the PETSc KSP suite of scalable linear equation solvers, which offers a range of solvers and preconditioners for solving sets of linear equations in parallel.

5.8 Input and output

High-level input for the new simulator is in the form of a text file in JSON format. JSON (http://www.json.org) is a lightweight open standard data-interchange format, often used for storing software configuration data. It can represent simple data types or hierarchies of more complex objects. It can be read and edited manually via a text editor or manipulated in code via libraries available for most programming languages. For parsing JSON input from Fortran our code uses the open-source FSON library (https://github.com/josephalevin/fson). Using JSON for input avoids many of the problems associated with a custom fixed-format file such as that used by TOUGH2, as well as the need to write any parsing code.

Particularly for large problems, it is desirable to be able to read some data from auxiliary files specified in the main input, possibly in other formats. For example, mesh data can be read in from separate files in mesh formats supported by PETSc's DMPlex class (currently ExodusII and GMSH).

For output, we plan to support multiple different formats including VTK (used for 3D visualization) and HDF5, the latest version of the Hierarchical Data Format. This is a standardized format designed to store and organize large datasets efficiently, and PETSc has built-in ability to read and write HDF5.

6. DEMONSTRATION PROBLEM

We present here results from running the new simulator on a simple test problem, to demonstrate some of its current capabilities. The model domain for the problem is a one-dimensional vertical column, 1 km deep, starting from isothermal hydrostatic conditions at a temperature of 20°C. Atmospheric pressure and temperature boundary conditions are applied at the top, and hot water with enthalpy 376.9 kJ/kg (corresponding to approximately 90°C temperature) is injected at 10 kg/s at the bottom.

Here we calculate the changing pressure and temperature profiles for this problem on a simple grid of 10 cells, with side length 100 m in each direction, using both the new code and TOUGH2 for comparison. For our code, we selected the IFC-67 thermodynamic formulation and backward Euler time stepping, for consistency with TOUGH2.

The initial hydrostatic conditions for both simulators are computed using TOUGH2, by removing the source term and running to a steady state (time 10^{14} s). Figures 1 and 2 show the modelled pressures and temperatures for the injection simulation at times 10^7 s, 10^8 s and 10^9 s. The results from the two simulators are identical, within the precision limitations of the fixed-format TOUGH2 output, showing the column of initially cold water heating up as the hot water is injected, and the pressure profile altering as a result of the lower density of the warmed fluid.

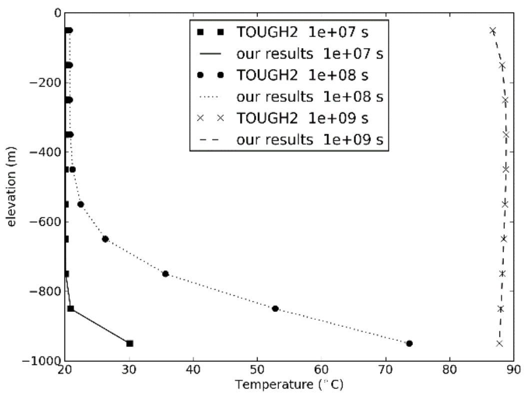

Figure 1: Modelled temperature vs. elevation for 1-D vertical hot-water injection problem at three different times, with TOUGH2 results shown for comparison

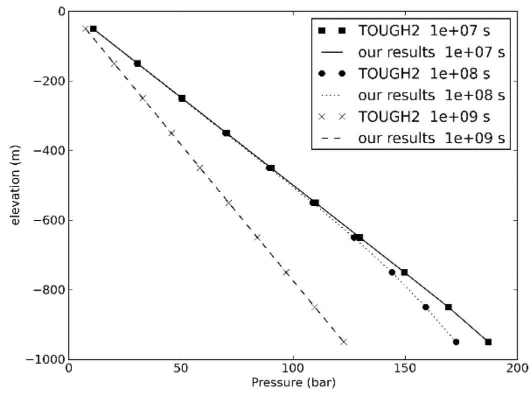

Figure 2: Modelled pressure vs. elevation for 1-D vertical hot-water injection problem at three different times, with TOUGH2 results shown for comparison

7. CONCLUSIONS

Most of the important components of the new flow simulator are now in place. It is able to simulate simple non-isothermal problems on 3-D unstructured meshes, running in serial or parallel, and produce results identical to those from TOUGH2 when comparable program options are selected. We are confident that the groundwork has been laid for a new simulator that meets the requirements for next-generation geothermal reservoir modelling.

ACKNOWLEDGEMENTS

The Geothermal Supermodels Programme is supported by a grant (C05X1306) from the NZ Ministry of Business, Innovation and Employment (MBIE), and by Contact Energy Ltd.

REFERENCES

Balay, S., Abhyankar, S., Adams, M., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Eijkhout, V., Gropp, B., Kaushik, D., Knepley, M., McInnes, L., Rupp, K., Smith, B., Zampini, S. and Zhang, H.: *PETSc Users Manual*, ANL 95/11, Revision 3.6, Argonne National Laboratory (2015).

Burnell, J., O'Sullivan, M., O'Sullivan, J., Kissling, W., Croucher, A., Pogacnik, J., Pearson, S., Caldwell, G., Ellis, S., Zarrouk, S. and Climo, M.: Geothermal Supermodels: the Next Generation of Integrated Geophysical, Chemical and Flow Simulation Modelling Tools. *Proc. World Geothermal Congress 2015*, Melbourne, Australia, 19-25 April (2015).

Croucher, A.E. and O'Sullivan, M.J.: Application of the computer code TOUGH2 to the simulation of supercritical conditions in geothermal systems. *Geothermics* 37, 622-634 (2008).

Flemisch, B., Fritz, J., Helmig, R., Niessner, J. and Wohlmuth, B.: DuMu^{x}: a multi-scale multi-physics toolbox for flow and transport processes in porous media. *ECCOMAS Thematic Conference on Multi-scale Computational Methods for Solids and Fluids*, Cachan, France, 28-30 November (2007).

Franz, P.: OOMPFS - a new software package for geothermal reservoir simulation. *Proc. World Geothermal Congress 2015*, Melbourne, Australia, 19-25 April (2015).

Kolditz, O., Bauer, S., Bilke, L., Bottcher, N., Delfs, J. O., Fischer, T., Gorke, U. J., Kalbacher, T., Kosakowski, G., McDermott, C. I., Park, C. H., Radu, F., Rink, K., Shao, H., Shao, H. B., Sun, F. , Sun, Y. Y., Singh, A. K., Taron, J., Walther, M., Wang, W., Watanabe, N., Wu, Y., Xie, M., Xu, W., Zehner, B.: OpenGeoSys: an open-source initiative for numerical simulation of thermo - hydro - mechanical/chemical (THM/C) processes in porous media. *Environ. Earth Sci.* 67, 589-599 (2012).

Mills, R.T., Lu, C., Lichtner, P.C. and Hammond, G.E.: Simulating subsurface flow and transport on ultrascale computers using PFLOTRAN. *Journal of Physics: Conference Series* 78 (2007).

Moridis, G.J., Kowalsky, M.B. and Pruess, K.: *TOUGH+HYDRATE v1.0 User's Manual: A code for the simulation of system behavior in hydrate-bearing geologic media*. Report LBNL-149E, Lawrence Berkeley National Laboratory, Berkeley, California, (2008).

O'Sullivan, J.P., Croucher, A.E., Yeh, A. and O'Sullivan, M.J.: Improved convergence for air-water and CO2-water TOUGH2 simulations. *Proc. 35th New Zealand Geothermal Workshop*, Rotorua, New Zealand, 17 - 20 November (2013).

Podgorney, R., Huang, H. and Gaston, D.: Massively parallel fully coupled implicit modeling of coupled thermal-hydrological-mechanical processes for enhanced geothermal system reservoirs. *Proc. 35th Stanford Geothermal Workshop*, Stanford University, Palo Alto, CA, 1-3 February (2010).

Pope, D.A: An exponential method of numerical integration of ordinary differential equations. *Numerical Analysis* 6(8), 491 - 493 (1963).

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).

Wagner, W., Cooper, J.R., Dittman, A., Kijima, J., Kretzschmar, H.-J., Kruse, A., Mares, R., Oguchi, K., Sato, H., Stöcker, I., Sifner, O., Takaishi, Y., Tanishita, I., Trübenbach, J., Willkommen, Th. : The IAPWS Industrial Formulation 1997 for the thermodynamic properties of water and steam. *ASME J. Eng. Gas Turbines Power* 122, 150–182 (2000).

Xu, T., Sonnenthal, E., Spycher, N. and Pruess, K. : TOUGHREACT- A simulation program for non-isothermal multiphase reactive geochemical transport in variably saturated geologic media: Applications to geothermal injectivity and CO2 geological sequestration. *Computers and Geosciences* 32, 145-165 (2006).

Yeh, A., Croucher, A. and O'Sullivan, M.J. : Recent developments in the AUTOUGH2 simulator. *Proc. TOUGH Symposium 2012*, Berkeley, California, September 17-19 (2012).

Zhang, K., Wu, Y., Ding, C. and Pruess, K. : TOUGH2_MP : a parallel version of TOUGH2. *Proc. TOUGH Symposium 2013*, Berkeley, California, May 12-14 (2013).

Zyvoloski, G.: *FEHM: A control volume finite element code for simulating subsurface multi-phase multi-fluid heat and mass transfer*. Report LAUR-07-3359, Los Alamos National Laboratory, Los Alamos, USA (2007).