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, a major part of which is the development of a new open-source, parallelised geothermal reservoir flow simulator.

This paper describes progress on flow simulator development over the past year, including the implementation of a modified multi-phase gravity term, parallel cell indexing, logging output in YAML format, improved boundary condition specification and automatic conversion of input from TOUGH2 format. Initial work has also been carried out on developing a comprehensive benchmark testing framework, integrated rock mechanics and the direct solution of natural-state problems without time-stepping. Investigations into improved linear equation solution for geothermal problems are also discussed.

Simulation results from two test cases (from the 1980 Geothermal Model Intercomparison Study) and the Wairakei reservoir model demonstrate the new simulator's ability to solve complex multi-phase geothermal flow problems efficiently in parallel.

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 (Burnell et al., 2015). The flow simulator software design, development workflow and initial development progress were described by Croucher et al. (2015). The present paper provides an update on flow simulator development progress over the past year. It also presents simulation results from two benchmark test problems and from a complex reservoir model (of the Wairakei geothermal field).

2. IMPLEMENTATION PROGRESS

2.1 Phase transitions and gravity term handling

Phase transitions using variable switching, for the pure water equation of state, are now fully implemented in the new simulator and tested.

In the AUTOUGH2 simulator (Yeh et al., 2012), phase transition behaviour was improved by calculating nominal fluid properties for phases not physically present. This can give a higher degree of flux continuity as cells change phase, and hence better convergence in the solution of the nonlinear mass and energy balance equations at each time step. This approach has also recently been incorporated into the iTOUGH2 simulator (Finsterle, 1999).

In the new simulator we have adopted a different approach to this problem. By implementing a modified form of the multi-phase gravity term in the flux calculation, taking into account the saturations of different phases, the same level of flux continuity can be achieved, but without the unnecessary computational expense of calculating absent phase properties.

Figure 1 shows simulation time step size histories for a simple 10-cell 1-D vertical column model, 1 km deep and with hot water injected at the bottom and atmospheric conditions applied at the top. This is the same as the model described in Croucher et al. (2015) but with hotter injected fluid (240°C), so that a steam zone develops in the upper cells. The model is run from cold initial conditions, using adaptive time stepping until a large time is reached, to approximate steady-state conditions.

Both AUTOUGH2 and the new simulator continue to a large simulated time (10^{15} s) with only a single time-step reduction at around 10^9 s (when most of the phase transitions occur). However the original TOUGH2 (version 2) simulator (Pruess, 2004), which has the original gravity term formulation and does not include absent phase calculations, reduces time step size twice at the same point, and eventually stops at 0.525×10^{12} s following further convergence trouble. Figure 2 shows the final temperature profiles for the three simulators, with good agreement between AUTOUGH2 and the new simulator but clearly indicating that the TOUGH2 simulation has not run long enough to achieve steady state.

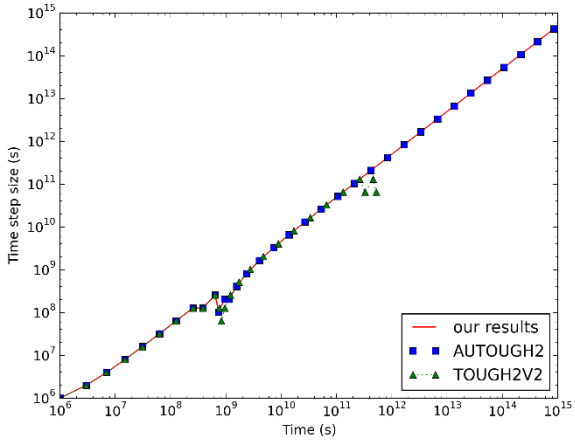

Figure 1: Time step size histories for running a vertical column model with a steam zone to steady state

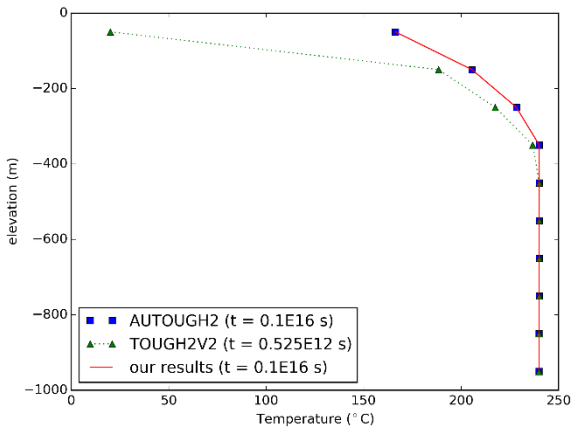

Figure 2: Final temperature profiles for vertical column model with a steam zone

2.2 Parallel cell indexing

Parallelization in the new simulator is implemented using the parallel data structures provided by PETSc (Balay et al., 2016) and domain decomposition to partition the model mesh amongst the processors. Each processor is assigned its own piece of the mesh and uses its own local mesh cell indexing.

When simulation results are output to file (in HDF5 format), each processor outputs the results for its own mesh partition. It would be possible to collate results from all processors and output them in the original "natural" (i.e. pre-partitioning) cell indexing, but this process would involve substantial parallel communication every time results were output (potentially every time step), thus reducing scalability.

However, it is sometimes necessary to be able to identify the natural index of each cell in the simulation output. This is needed, for example, when restarting a simulation from the results of a previous one, which was carried out using a different number of processors. In this case, the indexing of results will generally be different for the two simulations.

To address this problem, the new simulator now constructs and saves a separate natural index array in the output model results. When restarting from previous results, the index arrays for the previous and present runs are combined to create a "vector scatter" to assign initial conditions to the appropriate cells on the correct processors. This index array can also be used for post-processing of saved results.

2.3 Logging output

As a simulation runs it is useful to have "log messages" output to file and optionally to the display. These messages give feedback on events that occur during the simulation, for example, time stepping progress, non-linear solver progress during each time step, phase transitions, failure of linear or non-linear solvers, time step size reduction, or default initialization of variables not specified in the model input.

For the new simulator it was considered desirable to be able to parse logging output automatically, e.g. via a script, to enable automated diagnostics of model progress, termination, reasons for failure etc. This is particularly useful when running automated sequences of linked model runs (e.g. during model inversion), where a script may need to alter model input based on the progress of previous model runs.

To facilitate this, logging output is kept separate from the output of simulation results. As already mentioned, the main simulation results are output to HDF5 format, which is optimised for saving large tables of numerical results. Logging output is stored in a separate file in YAML format (http://www.yaml.org), a lightweight data serialization language that is easily human- and machine-readable. YAML has an advantage over e.g. JSON (which is used as the input format for the new simulator) in that the output should still be machine-readable even if the simulation crashes. YAML parsers are already available for many scripting and programming languages, obviating the need to write any specific log parsing code.

To help make parsing straightforward, all output log messages are written in a standardised layout with four components:

- **level**: describes the severity of the event, e.g. information, warning or error
- **source**: which part of the code the event occurred in, e.g. initialization, timestep, nonlinear solver
- **event**: what the event was, e.g. timestep start, nonlinear iteration, phase transition, timestep reduction
- **data**: any relevant data pertaining to the event, e.g. timestep size, iteration count

2.4 Rock mechanics

Substantial progress has been made on developing an integrated rock mechanics solver. This uses the same computational mesh as the flow solution and will be optionally called at each time step to update rock properties. Dirichlet boundary conditions have been implemented and the rock mechanics module has been run for simple preliminary problems- at this stage, only with linear elasticity, and not yet coupled to the flow solution.

2.5 Solution of natural-state problems

Calculation of natural-state conditions for a reservoir model is presently done by time-stepping the natural-state model from arbitrary initial conditions, using adaptive time-stepping until a large time-step size is reached. Although this process now works considerably better than it used to, owing to various improvements both to AUTOUGH2 and the new simulator (see section 2.1), there are still cases in which it is difficult to reach a large time-step size. This may result from non-convergence of the non-linear or linear solvers (see section 2.6).

Even if these problems are not encountered, time-stepping to natural state is often time-consuming. The computational effort involved in accurately calculating the entire time-evolution of the system is in a sense somewhat wasted, in that only the final result is actually used. Hence, alternative ways of reliably computing the natural state, not involving time-stepping, could give substantial computational savings. The flexible and modular code structure of the new simulator provides a convenient platform to explore alternative approaches.

The non-linear mass- and energy-balance equations solved at each time step of a transient simulation are of the form $d/dt\ \mathbf{L}(\mathbf{y}) = \mathbf{R}(\mathbf{y})$, where $\mathbf{y}$ is the vector of primary thermodynamic variables in the grid cells. $\mathbf{L}(\mathbf{y})$ represents the mass and energy balances in the cells, and $\mathbf{R}(\mathbf{y})$ represents inflows to the cells from fluxes through their faces, or from source and sink terms.

The simplest approach to solving the natural state problem is to attempt to solve $\mathbf{R}(\mathbf{y}) = \mathbf{0}$ directly, using a non-linear solver, starting from some initial estimate for $\mathbf{y}$. This naive approach has been implemented in the new simulator, but as expected, it generally does not work well unless the initial estimate for $\mathbf{y}$ is close to the correct solution. It may possibly still be of use for re-computing natural state solutions after only incremental changes in model parameters (e.g. during inverse modelling).

Further work in this area will focus on investigation of other approaches such as the method of "pseudo-transient continuation" (Kelley and Keyes, 1998), though this method would need some adaptation for the equations being solved here.

2.6 Linear equation solution

During each time step, a system of linear equations is solved for each non-linear solver iteration. These linear equations are frequently very ill-conditioned and hence difficult to solve. For natural state simulations, the ill-conditioning typically becomes progressively worse as natural state is approached. This is partly because in the non-linear equations, the left-hand side term $\mathbf{L}(\mathbf{y})$, which tends to increase the diagonal dominance of the linear equations, is progressively reduced.

In addition, certain features of the model configuration can increase ill-conditioning in the linear equations. Our experiments indicate that large contrasts in permeability across cell faces, even if present in only a single location, can result in increased ill-conditioning.

The new simulator uses the KSP suite of linear solvers provided by PETSc. Particularly for larger models, and in parallel, it is necessary to use iterative conjugate gradient methods, typically either GMRES or BiCGStab. We have found that the modified BiCGStab(L) (Sleijpen and Fokkema, 1993) linear solver can give improved performance as natural state is approached in some models.

It is also important to use an appropriate preconditioner when solving ill-conditioned problems. The incomplete LU (ILU) preconditioning used in AUTOUGH2 is effective for serial problems, but unfortunately does not scale well in parallel. Our experiments so far have shown that, for parallel problems, simple block Jacobi preconditioning can be sufficient for easier problems. For more difficult ones the "additive Schwarz" preconditioner sometimes gives better performance, but this depends also on the linear solver being used.

2.7 Automatic input conversion

A Python module has been developed for converting TOUGH2 model input to input files appropriate for the new simulator. This module relies heavily on the PyTOUGH scripting library for TOUGH2 (Croucher, 2011; Wellmann et al., 2012), and it is envisaged that it will eventually be incorporated into PyTOUGH itself. The module enables the user to automatically convert a MULgraph geometry file and TOUGH2 input data file into an ExodusII mesh file and JSON input file for the new simulator.

The main difficulty in converting input between the two simulators lies in the conversion of Dirichlet boundary conditions, which they treat in different ways. While TOUGH2 does this via boundary cells which must be explicitly included in the model mesh, the new simulator specifies boundary conditions on mesh faces (and internally adds "ghost cells" to them). To complicate matters further, while the mesh conversion process is able to retain the original cell indexing on the converted mesh (apart from that of the original boundary condition cells, which are removed), in general the mesh face indexing will be different (as this is done automatically by PETSc).

To make the conversion of boundary conditions simpler, a new way of specifying boundary faces was introduced. Previously this could only be done via mesh face indices. Now, a boundary face may optionally be specified instead by a cell and a direction vector. Here the cell is the one inside the model mesh boundary and the direction vector determines which face of that cell to apply the boundary condition to.

2.8 Benchmark testing framework

An automated framework is being developed for "benchmark testing", i.e. comparing simulation results with reference solutions for a variety of test problems. When complete, this will be integrated into the development workflow, so that these tests can be run automatically whenever significant changes are made to the code. This is distinct from unit testing (the testing of individual subroutines), which is already integrated.

The software framework for this is being based on CREDO, an open-source benchmark testing tool originally developed for the Underworld geodynamics simulator (Moresi et al., 2007). It provides Python scripting modules for the setup, execution and post-processing of tests, as well as summary output. While CREDO was designed in a relatively general (i.e. simulator-agnostic) way, it does contain some Underworld-specific aspects which we are adapting for use with our own simulator.

At the same time we are collating suitable benchmark and other test problems to include. Some of these have been taken or adapted from the 1980 Geothermal Model Intercomparison Study (Molloy, 1981) and are presented in section 3 below.

3. TEST PROBLEMS

Here we present results for two benchmark test problems taken from the 1980 Geothermal Model Intercomparison Study (Molloy, 1981).

3.1 Areal flow in 2-D reservoir

Problem 5 in the Model Intercomparison Study is a horizontal 2-D flow model, with regions of single-phase and two-phase flow (Pritchett, 1981). Convective heat transfer and phase transitions play important roles.

The model domain is rectangular with dimensions 300 × 200 m (and 100 m thick). The initial pressure is uniform at 36 bar, but the initial temperature varies according to a specified distribution (see figure 3). Pressure and

temperature are held constant at the right-hand boundary (x = 300 m), and fluid is withdrawn for 10 years at a rate of 5 kg/s from the production well at (x, y) = (62.5, 62.5) m. Corey's curves are used for relative permeability.

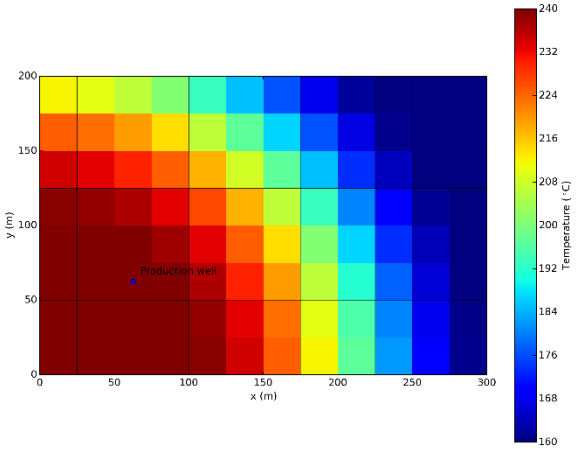

Figure 3: Initial temperature distribution for 2-D test problem

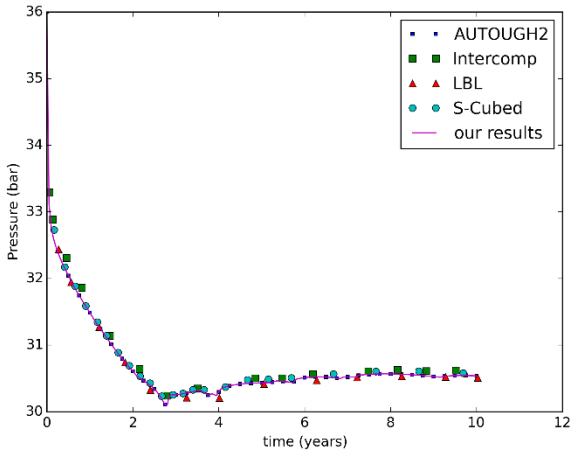

Figure 4: Pressure history in production well for 2-D test problem

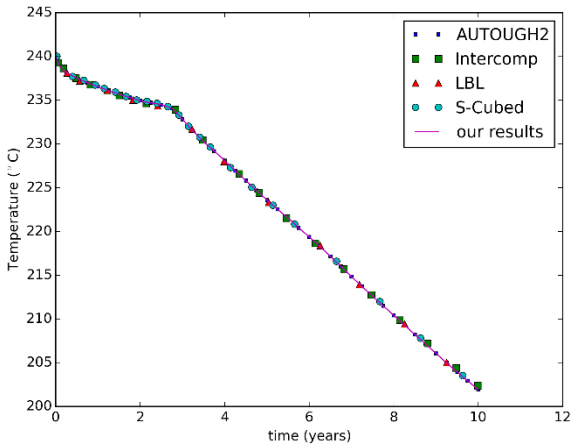

Figure 5: Temperature history in production well for 2-D test problem

Figures 4 and 5 show the pressure and temperature histories in the cell containing the production well, for the new simulator and AUTOUGH2, as well as three of the simulators (Intercomp, LBL and S-Cubed) used in the original Model Intercomparison Study. For this problem, six simulators were originally tested, but only these three achieved a reasonable consensus. The results from the new simulator and AUTOUGH2 are virtually identical, and both agree well with the older results.

Figure 6 shows the total mass of steam in place (per unit reservoir thickness) over time. Again, the results from the new simulator are virtually identical to those from AUTOUGH2, and also very close to the S-Cubed results. The LBL results are marginally higher (by approximately 1% at peak), and the Intercomp results higher again (by approximately 4% at peak).

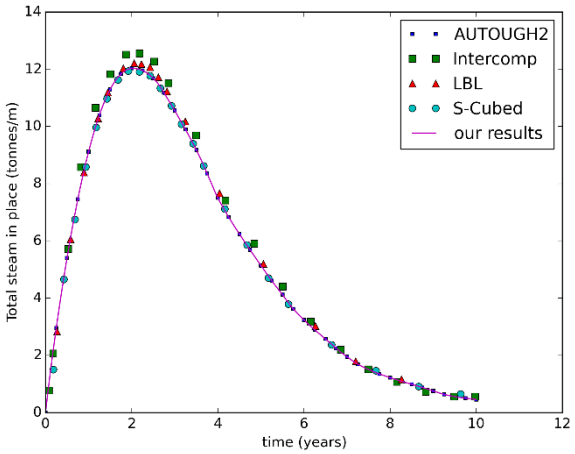

Figure 6: Total mass of steam in place over time for 2-D test problem

3.2 Flow in 3-D reservoir

Problem 6 in the Model Intercomparison Study is an idealised 3-D geothermal reservoir, with single-phase fluid at depth, overlain by a two-phase zone and a colder single-phase zone near the surface (Pruess, 1981). Production occurs below the steam zone at a stepped rate, increasing every two years. The model parameters are chosen so that boiling occurs at the well after approximately four years of production, and there is counter-flow between rising steam and the liquid water being drawn down towards the well. After six years, the increasing production rate causes the pressure in the well to approach zero.

The model grid, covering an area of 4 × 5 km and extending to a depth of 1.8 km, is very coarse by today's standards, containing only 125 cells. There are two rock types, with lower permeabilities assigned to the top and bottom layers and higher permeabilities in between. Corey's curves are used for relative permeability. The initial conditions have prescribed temperatures in all layers (saturations in the two-phase zone) and pressures assigned to approximate liquid hydrostatic conditions. Pressure and temperature boundary conditions are applied at the top and bottom of the model, and also along one lateral boundary.

Figures 7 and 8 show the modelled pressures and vapour saturations in the well block over the production period. The original study compared results from four simulators: LBL, Geotrans, S-Cubed and Intercomp. The first three of these were in good agreement, while the Intercomp results appeared to be in error. Results are shown here for the new

simulator and AUTOUGH2, together with LBL and S-Cubed.

Once again, there is a close match between the results from the new simulator, AUTOUGH2 and the older simulators. Vapour saturations from the new simulator are generally marginally lower than those from AUTOUGH2 and closer to those from S-Cubed. This is at least partly a result of the modified gravity term treatment (see section 2.1) which is expected to give slightly different solutions for problems involving gravity-driven flows in two-phase zones.

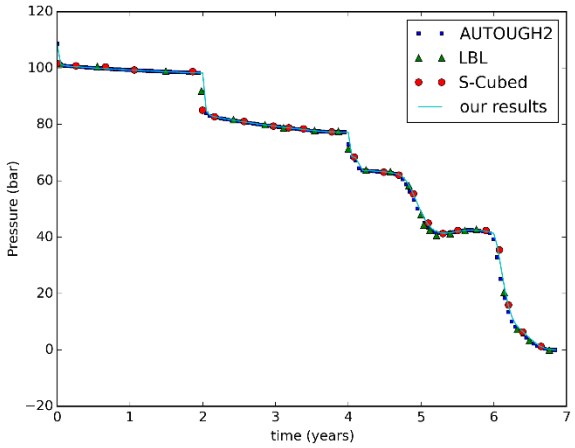

Figure 7: Pressure history in production well for 3-D test problem

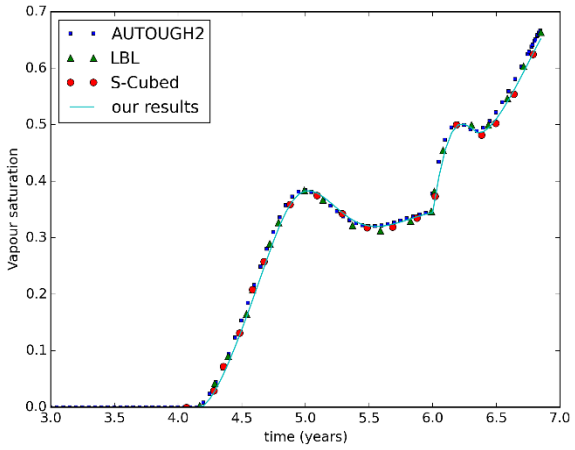

Figure 8: Vapour saturation history in production well for 3-D test problem

4. WAIRAKEI RESERVOIR MODEL

To demonstrate an application of the new simulator to a real and demanding geothermal reservoir model, we chose the version of the Wairakei AUTOUGH2 model described by Yeh et al. (2016). While previous versions of the Wairakei model (e.g. Yeh et al., 2014) used an irregular unstructured oval-shaped mesh, this one uses a regular rectangular mesh with approximately 31,000 cells. The mesh cells are 1 km × 1 km in the horizontal, and there are 38 layers extending to a depth of 4 km. It is a pure-water model with the top surface corresponding to the water table, on which atmospheric pressure and temperature boundary conditions are applied. Lateral boundaries are treated as zero-flux, while upflow boundary conditions are applied by distributing mass and heat generators over the bottom layer of the model.

The permeability structure is relatively complex: there are 80 rock types, with properties optimised using inverse modelling to match field data. The permeabilities of many rock types are significantly anisotropic and there are large permeability contrasts (up to five orders of magnitude) between some cells. As a result, the Jacobian matrix of the problem is extremely ill-conditioned and it is difficult to obtain a natural state solution.

The original mesh geometry and AUTOUGH2 input data file for the natural state model were converted to an ExodusII mesh and JSON input file for the new simulator using the Python module described above (section 2.7). For initial conditions, the AUTOUGH2 natural state results were used. These are a mixture of liquid and two-phase, and are close to the expected natural state for the new simulator, but not the same, because of the modified multi-phase gravity term. Hence, they provide a convenient starting point for investigation of the later stages of the natural state simulation, which are generally the most numerically difficult.

For this problem, natural state was considered to be reached at a target end time of 0.2×10^{16} s. The initial time step size was 0.2×10^{11} s. Running in parallel on six processor cores, natural state convergence was attained after 17 time steps, using the BiCGStab(L) linear solver and either the block Jacobi or additive Schwarz preconditioner. If the GMRES or standard BiCGStab linear solvers were used, with either preconditioner, the simulation would run to over 100 time steps without achieving natural state and with frequent linear solver failures and consequent time step size reductions.

The solution was compared with results from AUTOUGH2. A modified version of AUTOUGH2 was used, in which the changes to the multi-phase gravity term made in the new simulator (see section 2.1) had also been applied, to make the solutions more closely comparable. Starting from the same initial conditions, this version of AUTOUGH2 reached the target natural state end time in 23 time steps, using the BiCGStab linear solver and ILU preconditioner. Here it appears that the ILU preconditioner is able to compensate reasonably well for the shortcomings of the BiCGStab linear solver.

Figures 9 – 12 show the temperature profiles computed by AUTOUGH2 and the new simulator for four representative wells (WK270, WK321, TH 12 and NM 6), located in the Wairakei, Tauhara and Ngatamariki areas of the model. The agreement between the two simulators is very close for these profiles.

Figure 13 plots the temperatures from the new simulator over a vertical slice running NW – SE through the hottest part of the Wairakei – Tauhara system. Figure 14 plots the differences between these temperatures and those computed by AUTOUGH2 over the same slice. In general the differences are minor, though more noticeable (up to 0.05°C) in a few cells at the SE edge of the temperature plume, where horizontal temperature gradients are very high and even very small differences in the exact plume location can result in comparatively large differences in temperature in a particular cell. Over the whole model, temperature differences are again generally very small but

do reach 0.37°C at another cell on the NE edge of the Tauhara plume, near the TH 12 well. This well is difficult to calibrate against field data, again partly because of the high horizontal temperature gradients at the edge of the plume.

Figure 15 plots the vapour saturations from the new simulator over a vertical slice running NW – SE through the Rotokawa field, showing the modelled steam zone. Figure 16 plots the differences between these vapour saturations and those computed by AUTOUGH2 over the same slice. The largest differences are near the edge of the steam zone, but even these are very small. Over the whole model the maximum vapour saturation difference is of order 10^{-4}.

On a desktop PC with a 12-core Intel Xeon E5-2670 processor, AUTOUGH2 took approximately 437 s (wall clock time) to run this problem. The new simulator took approximately 353 s when run on one processor, and 171 s, 108 s and 76 s when run on two, four and six processors respectively. Of course, comparisons of this sort are problem-dependent. However, for this problem, the overall performance of the new simulator is comparable to AUTOUGH2 in serial, and scales reasonably well in parallel. PETSc documentation recommends problem sizes of at least 10,000 – 20,000 unknowns per processor for good parallel scaling, so this model (~ 60,000 unknowns) cannot be expected to scale well on more than about three processors. Larger models running on distributed compute clusters should demonstrate good scaling up to much larger processor counts.

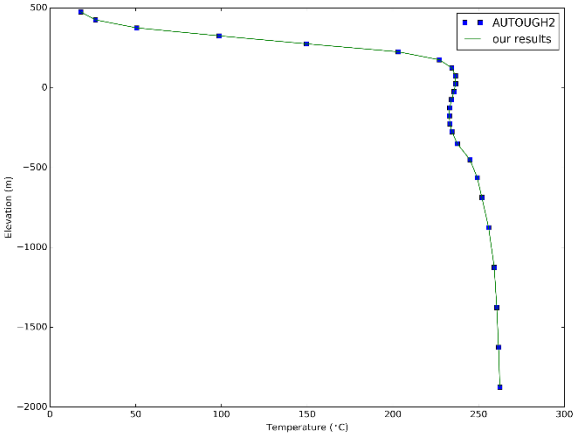

Figure 9: Modelled temperature profiles in Wairakei model for well WK270

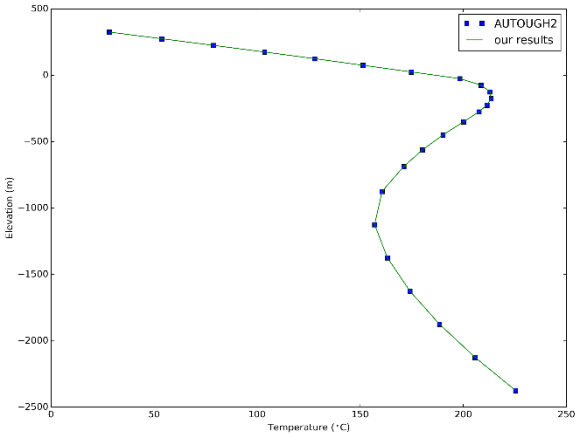

Figure 10: Modelled temperature profiles in Wairakei model for well WK321

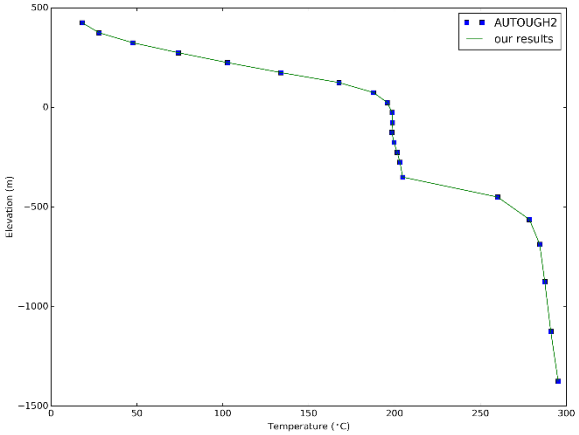

Figure 11: Modelled temperature profiles in Wairakei model for well TH 12

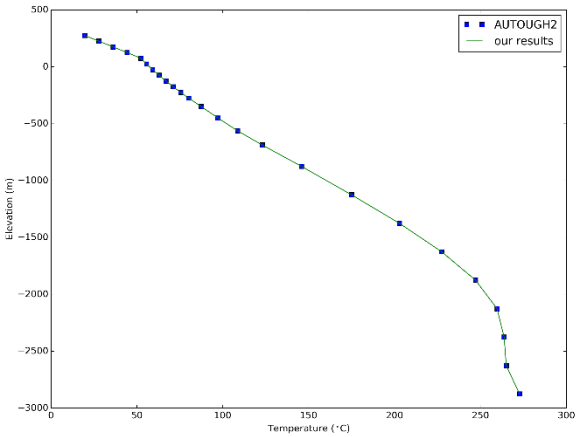

Figure 12: Modelled temperature profiles in Wairakei model for well NM 6

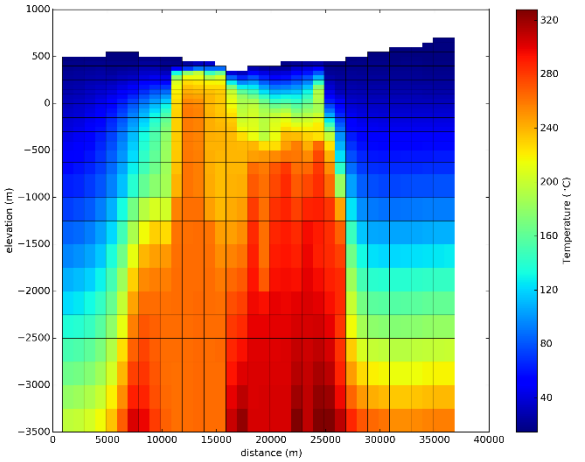

Figure 13: Temperatures computed by new simulator over NW – SE vertical slice through Wairakei – Tauhara system

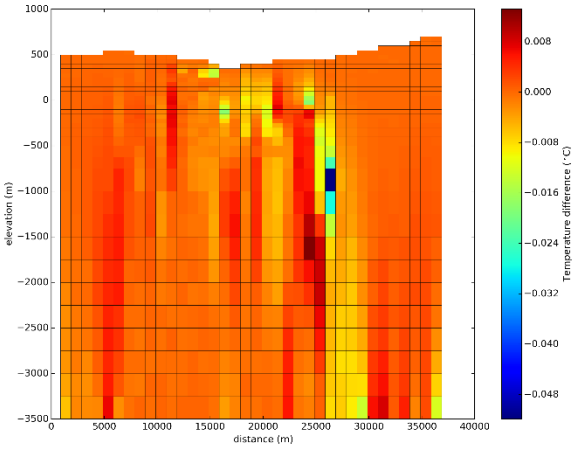

Figure 14: Temperature differences between new simulator and AUTOUGH2 over NW – SE vertical slice through Wairakei – Tauhara system

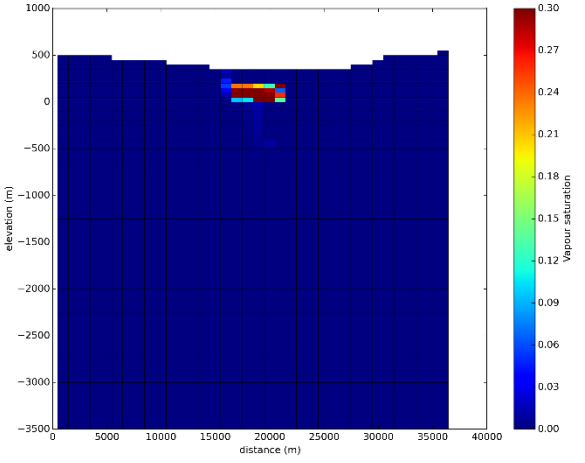

Figure 15: Vapour saturations computed by new simulator over NW – SE vertical slice through Rotokawa system

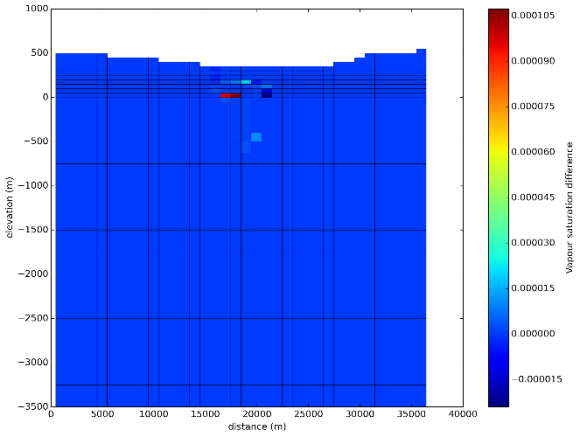

Figure 16: Vapour saturation differences between new simulator and AUTOUGH2 over NW – SE vertical slice through Rotokawa system

5. CONCLUSIONS

The past year has seen significant progress on implementation of the basic functionality of the new flow simulator, to the point where it is beginning to be usable for modelling real geothermal systems. It has also started to be useful from a research point of view for investigating alternative numerical techniques, e.g. for linear and non-linear solvers, and the solution of natural state problems. A substantial amount of effort has been devoted to testing, and so far this has given encouraging results both on benchmark test problems and in initial application to real geothermal models.

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., Karpeyev, D., Kaushik, D., Knepley, M., McInnes, L., Rupp, K., Smith, B., Zampini, S. and Zhang, H.: *PETSc Users Manual*, ANL 95/11, Revision 3.7, Argonne National Laboratory (2016).

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.: PyTOUGH: a Python scripting library for automating TOUGH2 simulations. *Proc. 33rd NZ Geothermal Workshop*, 21 – 23 November 2011, Auckland, New Zealand (2011).

Croucher, A.E., O'Sullivan, M.J., O'Sullivan, J.P., Pogacnik, J., Yeh, A., Burnell, J. and Kissling, W.: Geothermal Supermodels Project: an update on flow simulator development. *Proc. 37th NZ Geothermal Workshop*, 18 – 20 November 2015, Taupo, New Zealand (2015).

Finsterle, S.: *iTOUGH2 user's guide*. Lawrence Berkeley National Laboratory, LBNL-40040, 130 pp. (1999).

Kelley, C.T. and Keyes, D.E.: Convergence analysis of pseudo-transient continuation. *SIAM J. Numer. Anal.* 35(2), 508-523 (1998).

Molloy, M.W.: Geothermal reservoir engineering code comparison project. *Proc. Special Panel on Geothermal Model Intercomparison Study*, December 16-18, 1980, Stanford University, Stanford, California (1981).

Moresi, L., Quenette, S., Lemiale, V., Mériaux, C., Appelbe, B. and Mühlhaus, H.-B.: Computational approaches to studying non-linear dynamics of the crust and mantle. *Phys. Earth Planet. Inter.* 163, 69-82 (2007).

Pritchett, J.W.: The DOE code comparison project: summary of results for problem 5. *Proc. Special Panel on Geothermal Model Intercomparison Study*, December 16-18, 1980, Stanford University, Stanford, California (1981).

Pruess, K.: DOE project on geothermal reservoir engineering computer code comparison and validation: evaluation of results for problem 6. *Proc. Special Panel on Geothermal Model Intercomparison Study*, December 16-18, 1980, Stanford University, Stanford, California (1981).

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).

Sleijpen, G.L.G. and Fokkema, D.R: BiCGStab(L) for linear equations involving unsymmetric matrices with complex spectrum. *Elect. Trans. Numer. Anal.* 1, 11-32 (1993).

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).

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).

Yeh, A., O'Sullivan, M.J., Newson, J.A. and Mannington, W.I.: An update on numerical modelling of the Wairakei-Tauhara geothermal system. *Proc. 36th NZ Geothermal Workshop*, 24 – 26 November 2014, Auckland, New Zealand (2014).

Yeh, A., O'Sullivan, M.J., Newson, J.A. and Mannington, W.I.: Use of PEST for improving a computer model of Wairakei-Tauhara. *Proc. 38th NZ Geothermal Workshop*, 23 – 25 November 2016, Auckland, New Zealand (2016).