PROCEEDINGS, 43rd Workshop on Geothermal Reservoir Engineering
Stanford University, Stanford, California, February 12-14, 2018
SGP-TR-213

Benchmarking and experiments with Waiwera, a new geothermal simulator

Adrian Croucher, John O'Sullivan, Angus Yeh and Mike O'Sullivan

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

jp.osullivan@auckland.ac.nz

Keywords: reservoir models, numerical modelling, flow simulator, Waiwera

ABSTRACT

The "Geothermal Supermodels" project was a four-year New Zealand-based research programme, a major part of which was the development of a new open-source, parallelized geothermal reservoir flow simulator, called "Waiwera". This paper describes progress on Waiwera, presents the results of benchmarking tests and discusses experiments carried out on full-scale geothermal reservoir models.

Waiwera features 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. Thermodynamic behaviour has also been improved by implementing more sophisticated algorithms for phase transitions and for non-condensable gas energy of solution. Work has also been done to increase the efficiency of linear equation assembly and also solution, via improved pre-conditioning techniques.

The performance of Waiwera has been assessed using a suite of benchmark tests. The results are presented and the benchmark test models are included with the Waiwera installation. Waiwera has also been used to carry out a number of experiments on natural state simulations of full-scale geothermal reservoir models using different equations-of-state. Models of Ngawha (CO2-water) and an air-water system are presented and the findings discussed.

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), with a further report on implementation progress and test results presented by Croucher et al. (2016, 2017) and O'Sullivan et al. (2017).

The present paper provides a description of the flow simulator design and its development to date. It also presents simulation results from nine benchmark test problems and summary results from the application of Waiwera to two large-scale geothermal reservoir models.

1.1 Software Design

From the requirements laid out by Croucher et al. (2015), 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., 2016), 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 (Mills et al., 2007), which has demonstrated good scalability on very large simulation problems (up to 109 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. DuMu^x (Flemisch et al., 2007), OpenGeoSys (Kolditz et al., 2012) and OOMPFS (Franz, 2015) 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.

1.2 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.

2. FEATURES

2.1 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.

2.2 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.

The underlying geometric mesh Waiwera uses is useful for pre- and post-processing and necessary for rock mechanics. Previously only fully three-dimensional meshes were supported. However, more recently, support for radial and two-dimensional meshes has been added. Input specification of radial meshes is straightforward: an external mesh file is specified as usual, representing the two-dimensional $r - z$ mesh, together with a 'radial' flag. Waiwera then internally computes the cell volumes and face areas for the corresponding radial mesh. The $r - z$ mesh file can also be used for convenient post-processing and visualisation of output.

2.3 Sources and source controls

Geothermal reservoir models often require complex arrangements of sources and sinks, some of which may interact with each other and/or depend on reservoir conditions. This is particularly the case when modelling production runs and future scenarios. In developing the AUTOUGH2 geothermal simulator (Yeh et al., 2012) we added a variety of new generator types to address this problem.

A new and more modular approach has been developed for Waiwera. Instead of providing custom source types for specific situations, sources are kept simple and generic. However their parameters (principally flow rate and enthalpy) are managed by separate "source controls". Each source may have multiple source controls and these can be chained together to simulate complex behaviour. Alternatively, one source control may manage multiple sources simultaneously.

So far, the following source controls have been implemented: tables (for tables of flow rate or enthalpy vs. time), deliverability (for controlling flow rates based on a reference pressure), separators (for computing flow rates of separated steam and condensate) and limiters (for limiting flow rate based on a specified maximum total, steam or condensate flow).

These four source controls can be combined to mimic the behaviour of most of the AUTOUGH2 geothermal generator types. More types of source controls will be added as needed, and it is envisaged that these could also form the basis of a future surface network modelling system.

2.4 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.

2.5 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. PETSs' assembly of the finite-difference Jacobian matrix makes use of a "colouring" scheme to increase efficiency. This decreases the number of residual function evaluations needed, by determining which primary variables can be independently perturbed within a single residual function call (Balay et al., 2016).

In Waiwera the efficiency of this process has been increased further by re-using as much of the residual vector as possible between residual function calls, rather than re-calculating residuals that have not been affected by the perturbed primary variables. For the 31,000 cell Wairakei reservoir model described by Croucher et al. (2016) this resulted in run-time savings of approximately 25%. 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. Experiments using various combinations of solvers and preconditioners for full-scale geothermal reservoir models are discussed below.

2.6 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.

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.

2.7 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.

So far pure water EOS modules (isothermal and non-isothermal) have been developed as well as modules for simulating non-isothermal mixtures of water and non-condensable gases (NCGs). Within Waiwera's object-oriented code framework, this has been implemented as a generic NCG EOS from which specific EOS modules for particular gases can be derived. This approach maximises code re-use and consistency between different NCG modules.

All non-gas-specific code (e.g. for phase transitions) is kept in the generic NCG EOS module, so adding a module for a new gas essentially only involves implementing thermodynamic routines for the particular gas (for computing gas density, enthalpy, mixture viscosity and Henry's constant for gas dissolution into water).

For any NCG, the gas partial pressure is used as the additional primary thermodynamic variable. The energy of solution of the gas is calculated from the derivative of the logarithm of Henry's constant with respect to temperature (Himmelblau, 1959), the same approach used by the EWASG EOS module in TOUGH2 (Battistelli et al., 1997).

To date, specific NCG EOS modules have been implemented for the two most commonly simulated gases in geothermal modelling: CO2 and air. Like TOUGH2's EWASG EOS module, Waiwera's water/air EOS behaves slightly differently from TOUGH2's original water/air EOS modules (EOS3 and EOS4), owing to a more realistic thermodynamic formulation, including a temperature-dependent Henry's constant and a non-zero energy of solution for air. While the results are in most cases very similar to those obtained using TOUGH2 EOS3 or EOS4, our experiments indicate that this improved formulation can lead to better convergence in natural state models.

2.8 Dual porosity (MINC)

We have included simulation of fractured media in Waiwera via the Multiple INteracting Continua (MINC) method (Pruess and Narasimhan, 1985). The new Waiwera implementation allows users to specify multiple MINC zones in the model mesh with different parameters. All MINC processing of the mesh (i.e. adding cells representing flow in the matrix rock) is done internally by Waiwera, so users do not have to create a separate MINC mesh.

We plan to implement a method similar to that of Zyvoloski et al. (2008) for increasing execution speed of MINC models. This approach takes advantage of the particular sparsity structure of the linear equations arising from the MINC mesh topology, which is locally one-dimensional in the rock matrix, and effectively reduces the linear equation system size back down to that of a single-porosity model on the same mesh.

2.9 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.

2.10 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.

2.11 Numerical improvements

2.11.1 Non-dimensionalised primary variables

At each time step of a Waiwera simulation, a vector containing the updated solution is solved for. Previously this solution vector consisted of "raw" thermodynamic variables in each cell: fluid pressures and temperatures for single-phase water, pressures and saturations for two-phase water, with additional gas partial pressures for simulations involving NCGs.

These thermodynamic variables typically have very different magnitudes. This can cause problems when computing the finite-difference Jacobian matrix, which involves estimating a suitable increment for each variable. Waiwera uses the PETSc library (Balay et al., 2016) to carry out the Jacobian computation. If the variable v is large enough then PETSc takes the increment Δv as a fixed proportion (e.g. $\varepsilon = 10^{-8}$) of the variable itself ($\Delta v = \varepsilon v$).

However if the variable approaches zero (which can occur for saturations and partial pressures), this would no longer work, because the effects of the increment could become lost in round-off error. Hence, a cut-off value v_{min} is used, below which the increment is calculated instead as $\Delta v = \varepsilon v_{min}$. The problem then becomes one of choosing the appropriate cut-off value v_{min}. Selecting a single appropriate value is not possible if the variables have very different magnitudes.

To address this problem we have non-dimensionalised the Waiwera solution vector. Aside from finite differencing considerations, the PETSc documentation also recommends non-dimensionalisation as a way of improving non-linear solver convergence. The non-dimensionalisation used here is a simple scaling, carried out by dividing each variable by a fixed reference scale. The reference scales were determined by experiment to give the best overall performance on a range of different problems. By default, pressures (both fluid pressure and NCG partial pressure) are currently scaled by 10^6, while temperatures are scaled by 10^2 (saturations are already non-dimensional and of order unity). This has resulted in considerably improved performance on some problems, particularly simulations that include NCGs.

2.11.2 Phase transitions

Waiwera uses different primary variables to describe the fluid state in different thermodynamic regions, for example pressure and temperature in single-phase pure water, but pressure and vapour saturation for two-phase pure water. During the non-linear solver iterations at each time step, the fluid state may transition from one thermodynamic region to another (e.g. if there is a phase transition). In this case the thermodynamic trajectory of the fluid is temporarily stopped on the boundary between the two regions and restarted using the variables appropriate to the new region.

In the case of pure water, if the fluid boils the new two-phase variables are pressure and vapour saturation. The boundary vapour saturation is set just above zero. However there are various choices that could be made for the boundary pressure. If the old and new fluid states are (T_0, P_0) and (T_1, P_1) respectively (see Figure 1a), then previously Waiwera set the boundary pressure to $Ps(T_1)$, the saturation pressure at the new temperature. This is the same choice for the boundary pressure that is made in TOUGH2.

However this choice sometimes leads to transition failures and consequent time step reductions. As a result, we have modified Waiwera so that the boundary pressure is now taken from the intersection of the saturation curve $Ps(T)$ and the straight line between the old and new fluid states (P_x in Figure 1a).

Because the saturation curve is non-linear, we use a root-finding algorithm (Brent's method) to find the intersection point. The computational cost of this is relatively small (typically involving ~5 saturation curve function evaluations) and easily outweighed if any time step reductions are avoided.

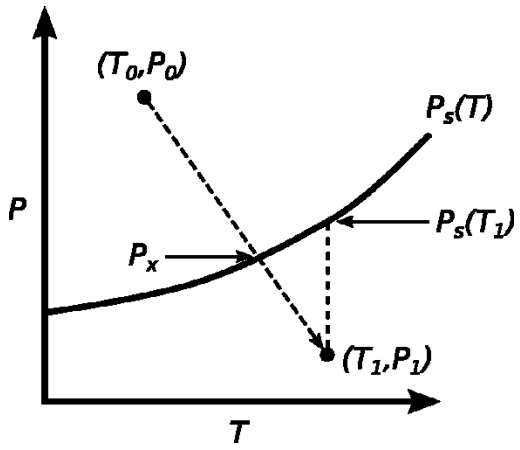
A very similar interpolation process is used to find the boundary pressure for transitions from dry steam to two-phase, and for transitions from two-phase to single-phase. The same technique is also applied to phase transitions with mixtures of pure water and NCGs.

The improved behaviour of the new phase transition scheme is illustrated in Figure 1b, which shows simulation time step size histories for the 10-cell 1-D vertical column model described by Croucher et al. (2016). This model starts from cold water initial conditions but develops a steam zone at around 10^9 s as a result of hot water (240°C) injected at the bottom. Previously Waiwera, like AUTOUGH2 (version 2.42), reduced time step at this point. TOUGH2 (v.2) reduces time step twice at this same point and later stalls at a time step size of around 10^{11} s. However, with the new phase transition scheme Waiwera now runs to a steady state at 10^{15} s with no time step reductions at all.

While a single time step reduction is of relatively little concern, running field-scale geothermal reservoir models to steady state often results in many phase transitions and time-step reductions. The new transition method does not eliminate these altogether, but it does make them significantly less frequent and usually enables reservoir models to reach steady state in many fewer time steps.

Waiwera also implements a modified form of the multi-phase gravity term in the flux calculation which takes into account the saturations of different phases that are present in different regions. This achieves a high level of flux continuity improving the phase transition process (Croucher et al., 2016).

a)



b)

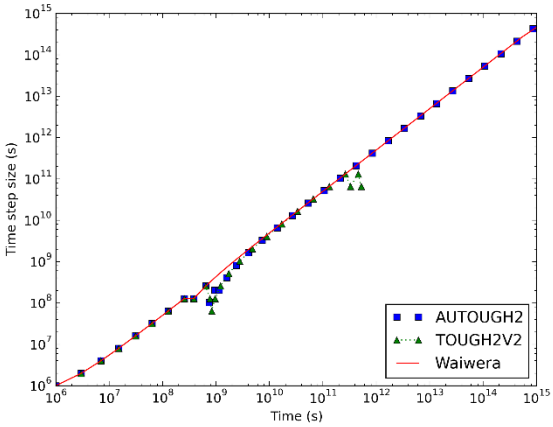


Figure 1: (a) Variable switching for boiling phase transition. (b) Time step size history for running a vertical column model with a steam zone to steady state.

3. BENCHMARK TESTS

3.1 Model Intercomparison Study problems

These benchmark tests are taken from the 1980 Geothermal Model Intercomparison Study (Molloy, 1981).

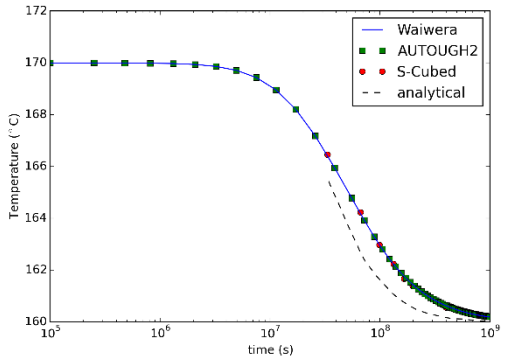
3.1.1 Problem 1: Avdonin model

This is a one-dimensional radial problem with steady single-phase flow and unsteady heat transport. Water at temperature 160°C is injected at 100 kg/s into a 170°C reservoir. An analytical solution (assuming constant fluid density, viscosity and heat capacity) was published by Avdonin (1964).

Figures 2a shows modelled temperature results vs. time at radius 37.5 m from the well. Results from the analytical solution as well as from AUTOUGH2 (v. 2.42) and S-Cubed (from the original study) are shown for comparison. There is very good agreement between the three numerical simulators, while the analytical results are slightly different. This is probably because they assume constant fluid properties whereas the numerical simulators more realistically allow these properties to vary with temperature and pressure.

Figure 2b shows the modelled temperatures vs. radius at time 10^9 s. Once again there is good agreement between the results from the three numerical simulators.

a)



b)

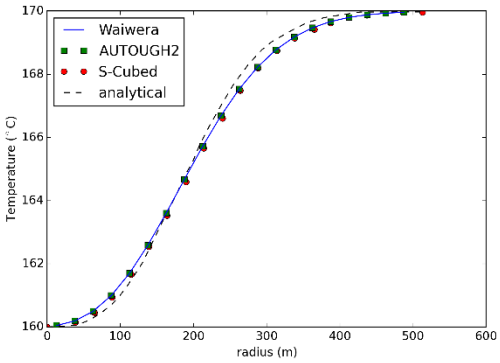


Figure 2: (a) Temperature vs. time at radius 37.5 m and (b) temperature vs. radius at time 10^9 s for Avdonin problem.

3.1.2 Problem 2: Radial well test

This is again a one-dimensional radial problem, but simulates production. There are three different cases: case A simulates single-phase flow, while cases B and C simulate two-phase flow. In case C a flashing front develops and propagates away from the well. For cases B and C a semi-analytical similarity solution is available (O'Sullivan, 1981). We simulated all three cases, but for the sake of brevity present only the results for the most demanding case C here.

We used a slightly modified version of the mesh specified in the original study. The original mesh specified the positions of the "nodes". However, it is not geometrically possible to construct a finite-volume mesh with cell centres at the specified nodal positions. The original mesh also used a relatively coarse discretisation near the well (the first cell having 0.5 m radial size). We used a mesh with slightly smaller innermost radial mesh sizes of 0.3, 0.4 and 0.6 m, and increasing logarithmically after that, with a total of 33 cells as opposed to the original 26.

Figures 3a and 3b show the pressure and liquid saturation solutions as a function of the similarity variable z = t / r2. Again, there is good agreement between the three numerical simulators, but they do not match the semi-analytical solution very well at larger values of z (e.g. near the well). Molloy (1981) suggested that the match might be improved if a finer mesh were used. Our mesh was slightly finer than that used in the original study, but this did not improve the match with the semi-analytical solution.

a) **b)**

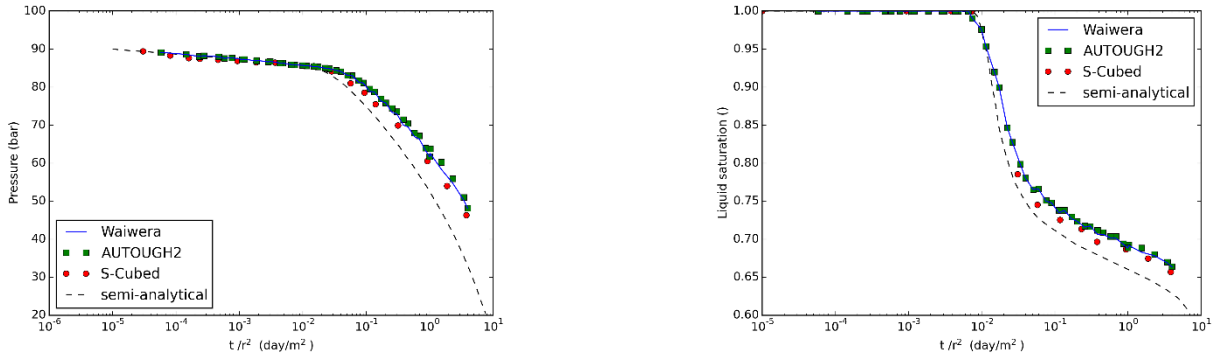


Figure 3: (a) Pressure solution for radial well test case C and (b) liquid saturation solution for radial well test case C.

3.1.3 Problem 4: Expanding two-phase zone

This is a one-dimensional vertical model, 2 km deep, with an initially hydrostatic water column disturbed by constant-rate mass withdrawal from the bottom over a 40-year period. The upper half of the model has lower permeability, and the initial temperature profile is piecewise linear with temperatures of 310°C at the bottom, 290°C in the middle and 10°C at the top. This problem is sufficiently complex that there is no analytical or semi-analytical solution available. Hence we compare the Waiwera results with those from the S-Cubed and AUTOUGH2 simulators.

Figures 4a and 4b show the modelled pressure and liquid saturation histories at depth 1550 m. Again the three numerical simulators give very consistent results. The agreement at other depths (not shown here) was similarly very good.

3.1.4 Problem 5: 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.

Figures 5a and 5b 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 5c 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).

a) **b)**

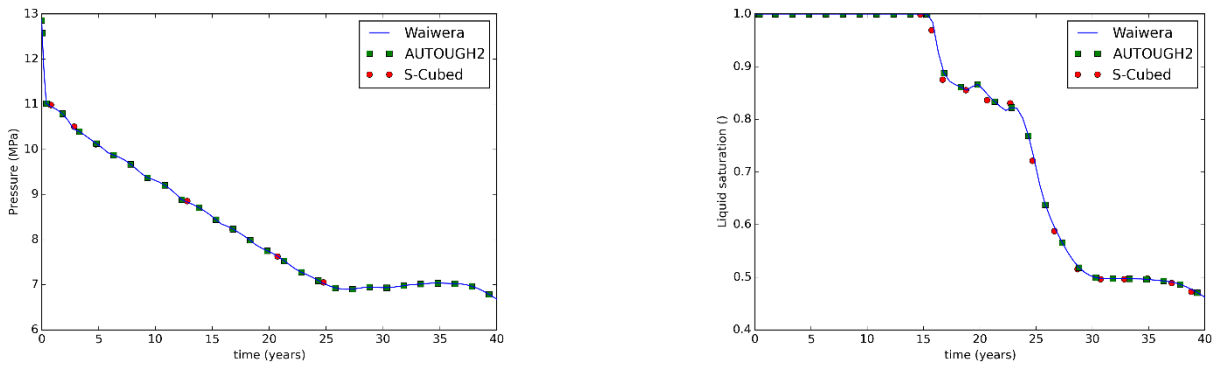


Figure 4: (a) Pressure history and (b) liquid saturation at depth 1550 m for expanding two-phase zone problem.

a) **b)** **c)**

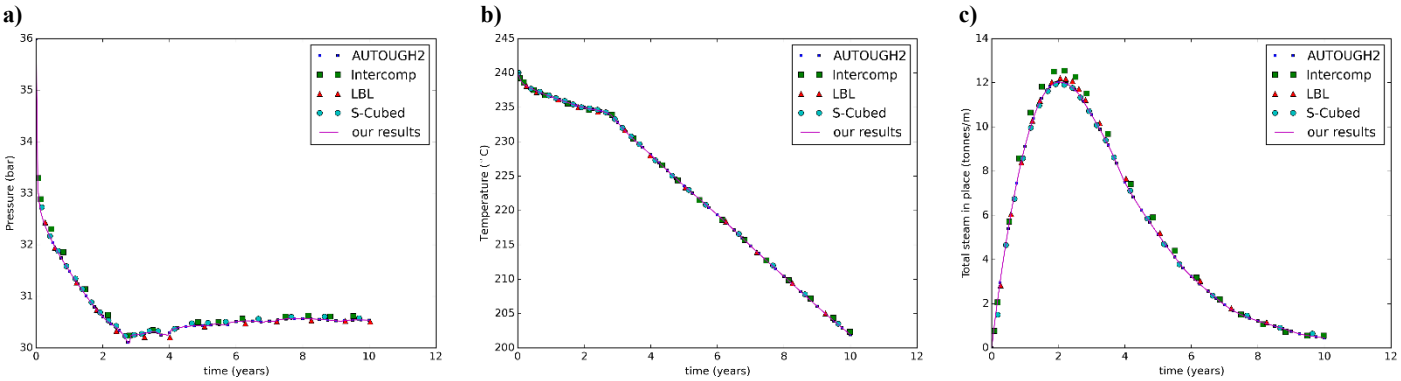


Figure 5: (a) Pressure history, (b) temperature history and (c) total mass of steam in place over time for 2-D reservoir problem.

3.1.5 Problem 6: 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 6a and 6b 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 2.11.2) which is expected to give slightly different solutions for problems involving gravity-driven flows in two-phase zones.

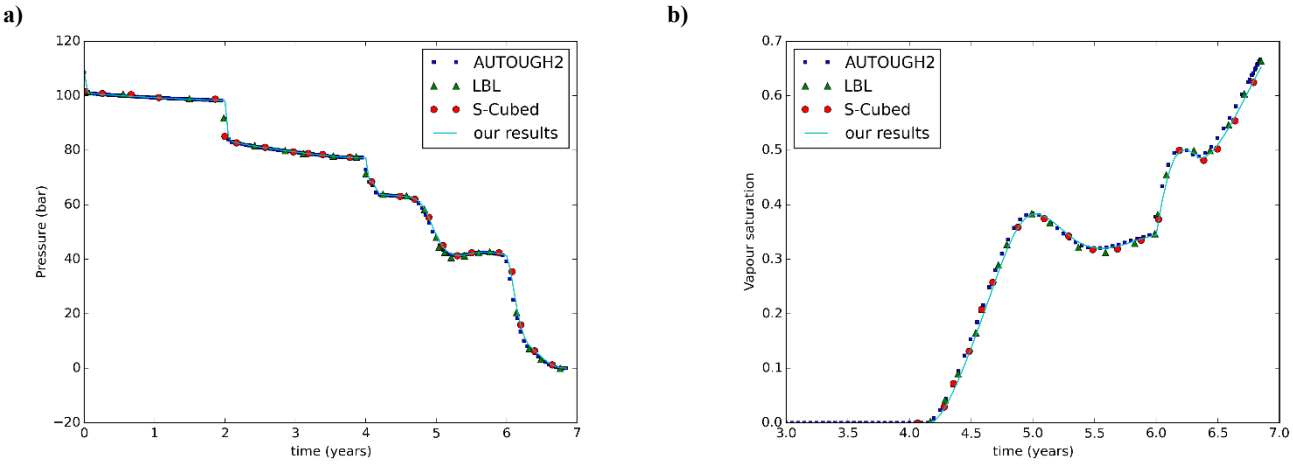


Figure 6: (a) Pressure history and (b) vapour saturation in production well for 3-D test problem.

3.2 Lumped-parameter CO_2 problem

This simple problem was described by O'Sullivan et al. (1985) and effectively simulates the effect of CO_2 on depletion from a lumped parameter model. It provides a simple but useful first benchmark for testing a simulator's water-CO_2 mixture thermodynamics.

The single 1 m^3 cell is initially at temperature 260°C, vapour saturation 0.2 and CO_2 partial pressure 30 bar. Fluid is produced at a constant rate of 5 kg/s. Two different sets of relative permeability curves were used to show their effect on the solution; for our benchmarking purposes we simulated only the case using the Corey curves.

The modelled pressure history is shown in Figure 7a, together with results from AUTOUGH2 and the original MULKOM results. Once again there is excellent agreement between Waiwera and the other numerical simulators.

3.3 CO2 column problem

For a more complex CO_2 test we used the vertical column steady-state model from O'Sullivan et al. (1985). The column is 1 km deep with a caprock zone (permeability 0.5 mD) extending to a depth of 300 m, below which is a reservoir zone with permeability 20 mD. Atmospheric conditions at 10°C are prescribed at the top, and heat and mass are injected at the bottom (2 kg/s per km^2, with enthalpy 1300 kJ/kg). The model is run from isothermal hydrostatic initial conditions (with no CO_2) to a steady state.

O'Sullivan et al. (1985) presented results from the MULKOM simulator for five cases with increasing CO_2 content in the injected fluid. However it is not entirely clear how much CO_2 was injected in each case. Their figures 10 and 11 indicate that the cases corresponded to partial pressures of CO_2 of 0, 0.1, 1, 5 and 20 bar, but the text suggests that these values may have referred to percentage mass fractions of injected CO_2. The reported partial pressures at the bottom of the model are also not consistent with the labelled values of input partial pressure.

Trying the simulations with the labelled values interpreted as partial pressures, we found that the model would not run to steady state on either AUTOUGH2 or Waiwera. Interpreting them as percentage mass fractions, the model would run to steady state (except for the 20% case), and while the CO_2 partial pressures matched the original MULKOM results reasonably well, the vapour saturations were considerably lower than those from MULKOM (by a factor of around 2 for the 5% case). This suggests there may be further uncertainty around the reported model parameters.

We ran the model using the specified linear relative permeability functions (from figure 4 in the original paper) and injected CO_2 mass fractions of 0.1%, 1% and 5%. Figure 7b shows the modelled steady-state CO_2 partial pressures vs. elevation for each of the three cases. There is excellent agreement between the Waiwera and AUTOUGH2 results, and the match with the original MULKOM results is reasonable considering the uncertainty around the input parameters. O'Sullivan et al. (1985) also did not give any details of their model mesh. We experimented with several meshes and found this had a noticeable effect on the solution. The results shown used a mesh with 30 m cells in the caprock zone and 35 m cells in the reservoir.

To run the 0.1%, 1% and 5% cases to steady state ($t = 10^{15}$ s), Waiwera took 44, 51 and 87 time steps respectively. The corresponding AUTOUGH2 (v. 2.42) simulations took 72, 97 and 198 time steps, i.e. approximately twice as many in each case. Waiwera's more rapid steady-state convergence for this model is due mainly to the modified phase transition algorithm (section 2.2.2).

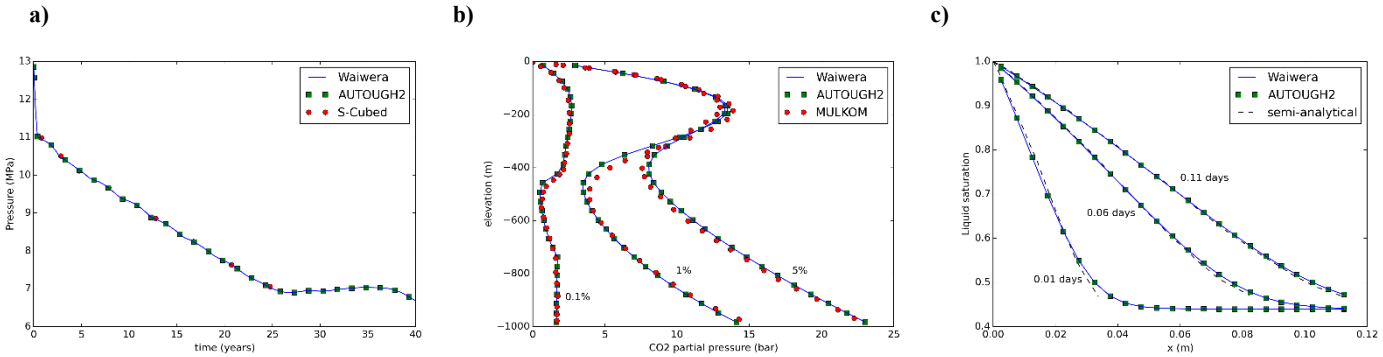


Figure 7: (a) Pressure history for lumped-parameter CO_2 problem. (b) CO_2 partial pressure vs. elevation for CO_2 column problem. (c) Liquid saturation vs. distance for 1-D infiltration problem.

3.4 One-dimensional infiltration problem

This problem, taken from the TOUGH user's guide (Pruess, 1987) is an isothermal air-water problem simulating infiltration into a horizontal tube of partially saturated soil. A semi-analytical solution is available.

The model mesh is only 20 cm long, with 0.5 cm cells. Pressure and temperature boundary conditions are applied at the left-hand end, and the initial vapour saturation is specified as 0.56 inside the model domain. Linear relative permeability and capillary pressure functions are used.

Figure 7c plots the modelled liquid saturations vs. distance along the tube at times 0.01, 0.06 and 0.11 days. The Waiwera results match both the AUTOUGH2 results and the semi-analytical solution (digitised from figure 7 in Pruess (1987)) very closely.

3.5 Heat pipe problem

This is the "radial heat pipe" problem from the TOUGH2 user guide (Pruess et al., 1999), which simulates a cylindrical 3 kW heater in a porous medium containing a mixture of water, steam and air. A 120-cell one-dimensional radial mesh is used, with logarithmically increasing cell sizes, and fluid conditions are held constant at the boundary (r = 10 km). Van Genuchten relative permeability and capillary pressure functions are used.

Figure 8 shows the modelled temperature, vapour saturation and vapour air mass fraction results vs. radius after 10 years. The Waiwera results are practically indistinguishable from the results from AUTOUGH2 and TOUGH2 (digitised from figure 19 in Pruess et al. (1999)).

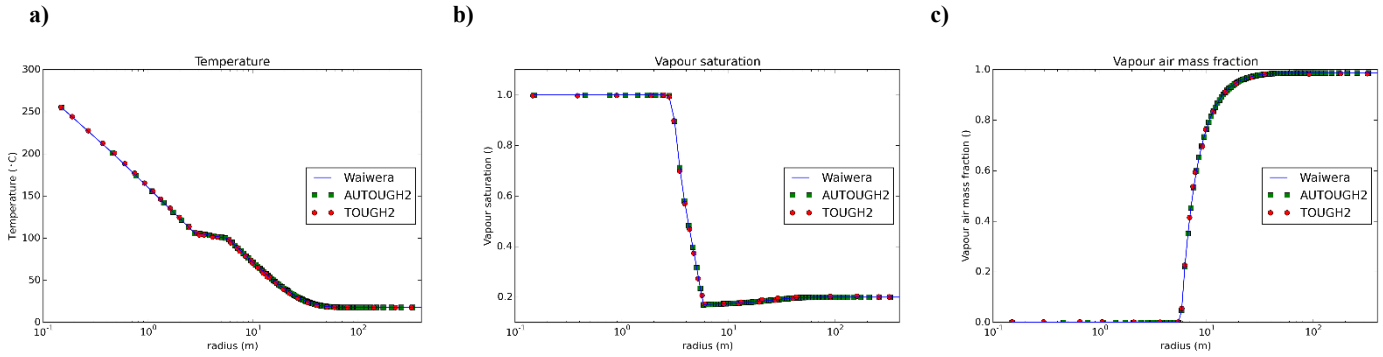


Figure 8: (a) Temperature, (b) vapour saturation and (c) vapour air mass fraction vs. radius after 10 years for radial heat pipe problem.

3. FULL-SCALE SIMULATIONS

The present paper discusses the application of Waiwera to simulations of the natural state of two geothermal systems, namely Ngawha (New Zealand) and System B. These two models were selected because they both run poorly with a standard version of TOUGH2, i.e. they do not converge well to a stable steady state. Even after many time steps the size of the time step and the total time are not as large

as desired. A well-performing natural state model will achieve a time step of 10^{13} -10^{14} s and a total time of 10^{15} - 10^{16} s in less than 500 time steps. With a standard version of TOUGH2 both the Ngawha and System B models struggle to achieve this.

The features of the models of Ngawha and System B that make them run slowly are: they both involve mixtures of water and a non-condensable gas (carbon dioxide for Ngawha and air for System B) and they both have large contrasts in permeability. We have found with other models that these characteristics can give poor convergence to a natural state.

3.1 Ngawha Model

The Ngawha model has 1015 cells per layer and 32 layers, with some of the shallow layers being incomplete as the top of the model follows the topography), giving a total number of 29,590 cells. A CO2/water equation of state (EOS) is used because several of the wells produce ~1% mass fraction of CO2. In this version of the model the top is set at the estimated position of the water table, i.e., the vadose zone is not included in the model. More details of the Ngawha model setup can be found in O'Sullivan et al. (2017).

This particular version of the Ngawha model was selected because of its poor numerical behaviour with TOUGH2, thus providing a severe test of simulator performance. From a physical perspective the model has some shortcomings, with some of the temperatures at the base of the model being too high, up to 336°C. (Strictly speaking, these temperatures are slightly outside the range of applicability for some of the CO_2 thermodynamic functions used in TOUGH2 and Waiwera, such as the Henry's constant equation which is valid only up to 300°C.)

Simulations were carried out with TOUGH2, AUTOUGH2 and Waiwera on a desktop PC with a 12-core Intel Xeon E5-2670 processor. Natural state was considered to be achieved at time 1E+15 s.

Table 1 shows the performance of the TOUGH2 and AUTOUGH2 simulators on this problem. The TOUGH2 run failed after only 6 time steps, with an unrecoverable error in the linear solver. Other linear solvers such as Bi-CGSTAB(*l*) (Sleijpen and Fokkema, 1993) gave similar behaviour. AUTOUGH2 did manage to reach a natural state solution, but it took nearly 400 time steps and over 8.5 hours of run time.

Table 2 shows Waiwera's performance on this problem, for various combinations of linear solvers, preconditioners and numbers of processors (np). In each case, as well as the number of time steps and total run time, the number of linear solver failures is given. For all cases shown, the ILU(0) sub-preconditioner was used.

Our initial experiments used the Bi-CGSTAB(*l*) linear solver and Additive Schwarz (ASM) preconditioner, as this had been an effective combination for other models (e.g. Wairakei), usually giving better performance than GMRES (Saad and Schultz, 1986) or BiCGSTAB (Croucher et al., 2016). For the Ngawha model this gave natural state run times of approximately 1 – 1.5 hours (depending on the number of processors), i.e. about 6 - 7 times faster than AUTOUGH2, and with fewer time steps but with the increased speed mainly due to the parallelisation. However, the linear solver performance was still not very good, with typically over 100 solver failures per run. Large numbers of failures also often mean that even when the linear solver does converge, it does so slowly. Here the number of failures generally increased with the number of processors, so that the run times were not decreasing as the number of processors increased.

Switching to the Block Jacobi preconditioner improved the performance considerably, with solution times for 8 and 10 processors down to less than 15 minutes and with far fewer solver failures. Unusually, the 6-processor run performed even better, with a 3.5 minute run time, no solver failures and only 28 time steps.

Following the advice of the PETSc documentation, we then tried using the restarted GMRES linear solver, and experimented with various restart intervals. Our experiments indicated that restarting after 200 iterations gave the best results for this problem, so only results for GMRES(200) are shown here. For GMRES(200) and the ASM preconditioner, the run times were generally 1 – 3 times faster than those for BiCGSTAB(*l*) with the Block Jacobi preconditioner (except when running on 6 cores where the latter combination performed unusually well). However, the best results of all were obtained by combining GMRES(200) with the Block Jacobi preconditioner, particularly for 8 and 10 processors where there were no linear solver failures and the run times were under 2 minutes. For both preconditioners, the GMRES(200) solver performed better as the number of processors increased, with fewer failures and lower time step counts (which was not generally the case for the BiCGSTAB(*l*) solver).

For the cases where linear solver failures still occurred, we also experimented with increasing the level of ILU sub-preconditioner fill-in. However this did not appear to reduce the instance of failures. For this model, the most effective way to achieve good performance was by using the restarted GMRES(200) linear solver.

This problem is clearly extremely ill-conditioned, as the convergence results are very sensitive to the details of the linear solver and preconditioner. However, Waiwera was able to produce a converged natural state solution up to 200 times faster than AUTOUGH2.

3.1 System B Model

The System B model has 2496 cells per layer and 77 layers, with some of the shallow layers being incomplete as the top of the model follows the topography (on land) and the bathymetry (under the ocean), giving a total number of cells of 127485. An air/water equation of state (EOS) is used as it is important to accurately represent the shallow zone including the unsaturated, vadose zone. More details of the model setup can be found in O'Sullivan et al. (2017).

Table 1: TOUGH2 / AUTOUGH2 natural state simulations for Ngawha

Simulator	Linear solver	Time steps	Run time (min)
TOUGH2	Bi-CGSTAB	-	Run failed
AUTOUGH2 (v 2.42)	Bi-CGSTAB	397	518

Table 2: Waiwera natural state simulations for Ngawha

Linear solver	Precon-ditioner	np	Time steps	Run time (min)	Solver failures
Bi-CG-STAB(*l*)	ASM	4	249	78.9	95
		6	278	72.7	107
		8	318	81.1	125
		10	299	91.0	117
	Block Jacobi	4	252	69.1	96
		6	28	3.5	0
		8	67	13.2	16
		10	42	8.7	6
GMRES (200)	ASM	4	82	23.2	10
		6	52	9.0	5
		8	51	10.4	4
		10	34	4.7	1
	Block Jacobi	4	50	10.5	4
		6	41	4.0	1
		8	33	2.0	0
		10	31	1.9	0

Table 3: TOUGH2 / AUTOUGH2 natural state simulations for System B

Simulator	Linear solver	Time steps	Max. time step	Run time (min)
TOUGH2	Bi-CG, Lanczos-type PC	500	1.51×10^{13}	1052
AUTOUGH2 (v 2.42)	Bi-CGSTAB	500	2.20×10^{13}	1106

Table 4: Waiwera natural state simulations for System B

Linear solver	Precon-ditioner	np	Time steps	Run time (min)	Solver failures
GMRES(*l*) (200/10)	ASM	8	405	141.2	6
		16	381	63.9	2
		32	363	50.9	1
		64	497	203.6	28
	Block Jacobi	8	403	209.6	4
		16	416	126.3	11
		32	376	86.9	10
		64	778	149.3	13

Initial conditions were obtained by interpolating results from a coarser model and then natural state simulations were carried out with TOUGH2, AUTOUGH2 and Waiwera with different choices of preconditioner. The model was run either to convergence or to a total of 500 time steps. The results for TOUGH2 and AUTOUGH2 are given in Table 3 and the results for Waiwera in Table 4.

For TOUGH2 and AUTOUGH2 simulations were carried out on a desktop PC with an Intel Xeon E5-2670 processor. The Waiwera simulations were carried out on the NeSI Pan Cluster at the University of Auckland. They were submitted to the Sandy Bridge architecture cores which have Intel E5 2680 processors and 16 cores per node. The internal network uses QDR Infiniband at 40Gb/s. Natural state was considered to be achieved at time 1E+15 s.

Table 3 shows that neither TOUGH2 nor AUTOUGH2 achieved the natural state target time within 500 time steps. In both cases the linear solves become very expensive as the time step size increases and the system of equations become more ill-conditioned. TOUGH2's time step size plateaued at approximately 1.5×10^{13} seconds and when the simulation reached 500 time steps it had only achieved a total time of 1.43×10^{14} seconds, 14% of the total target. Using the default linear solver settings in TOUGH2 equates to the Bi-CG Lanczos-type preconditioned method with the number of iterations in the linear solver limited to 10% of the number of equations (~500,000 for this

model). Thus the linear solver was allowed up to 50000 iteration to obtain a solution resulting in no linear solver failures but very long solution times. Various options could be applied to attempt to optimise the solution process though at over 17 hours to achieve the first 500 timesteps even the most effective combination of settings is likely to result in excess of two days to achieve a converged natural state.

The results for AUTOUGH2 were quite similar to those obtained by TOUGH2. AUTOUGH2 achieved a slightly higher maximum time step and as a result achieved a total time of 2.32×10^{14} seconds, 23% of the total target. However the larger time steps required more effect in the linear solver and the run time to achieve 500 time steps was longer at over 18 hours.

A number of different combinations of linear solvers and pre-conditioners were tested for the Waiwera simulations of the System B model using a range of different numbers of processors. Following on from the results of Ngawha model GMRES was generally found to more effective than Bi-CGSTAB and Bi-CGSTAB(*l*). Also the accelerated variant GMRES(*l*) (Baker et. al., 2005) was found to perform better than the standard GMRES solver and the restart parameter of 200 was found to be effective. As such, detailed timing runs and comparisons of two different types of pre-conditioners were only run using GMRES(*l*).

Table 4 shows that unlike the Ngawha model, the ASM pre-conditioner outperforms the Block Jacobi pre-conditioner. It also shows that in both cases increasing the number of processors from 32 to 64 results in significant increases in the run times. The number of linear solver failures increases as well which not only increases the run times but the number of time steps required to achieve a converged solution. For both the ASM and Block Jacobi pre-conditioners the speedup is good with an increase from eight to 16 processors. Increasing to 32 processor offers smaller gains in both cases.

Comparing the results between Waiwera and TOUGH2 and AUTOUGH2 shows that not only does Waiwera scale well for the System B model but its better implementation of the solution algorithm combined with more effective linear solver options allow a converged solution to be achieved much more rapidly. Results are now available with an hour as opposed to requiring days making calibration of this large model now possible.

3. CONCLUSIONS

The design, development and features of the new geothermal simulator Waiwera have been presented. Results from a number of benchmark problems from the 1980 Geothermal Model Intercomparison Study show that simulator performs very well. Several other benchmark test have been carried out with very good results and are they now also included as part of the Waiwera software. Experiments on large-scale, real geothermal reservoir models have shown that Waiwera scales well and outperforms industry standard geothermal simulators.

We are planning for the first Waiwera 1.0 open-source release later this year.

3. 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

Avdonin, N.A.: Some formulas for calculating the temperature field of a stratum subject to thermal injection. *Meft'i Gaz* **3**, 37 – 41 (1964).

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).

Baker, A.H., Jessup, E.R., and Manteuffel, T.A.: A technique for accelerating the convergence of restarted GMRES. *SIAM Journal on Matrix Analysis and Applications*, **26** (2005)

Battistelli, A., Calore, C. and Pruess, K.: The simulator TOUGH2/EWASG for modelling geothermal reservoirs with brines and non-condensible gas. *Geothermics* **26(4)**, 437 – 464 (1997).

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).

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. 38th NZ Geothermal Workshop*, 23 – 25 November 2016, Auckland, New Zealand (2016).

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. 39th NZ Geothermal Workshop*, 22 – 24 November 2017, Rotorua, New Zealand (2017).

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).

Himmelblau, D.M.: Partial molal heats and entropies of solution for gases dissolved in water from the freezing to near the critical point. *J. Phys. Chem.* **63(11)**, 1803 – 1808 (1959).

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).

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).

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).

O'Sullivan, J.P., Croucher, A.E., Yeh, A., and O'Sullivan, M.J.: Experiments with Waiwera, a new geothermal simulator. *Proc. 39th NZ Geothermal Workshop*, 22 – 24 November 2017, Rotorua, New Zealand (2017).

O'Sullivan, M.J.: A similarity method for geothermal well test analysis. *Water Resources Res.* **17(2)**, 390 – 398 (1981).

O'Sullivan, M.J, Bodvarsson, G.S., Pruess, K. and Blakeley, M.R.: Fluid and heat flow in gas-rich geothermal reservoirs. *Soc. Pet. Eng. J.*, April 1985, 215 – 226 (1985)

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

Pruess, K.: TOUGH user's guide. LBL-20700, Lawrence Berkeley National Laboratory, Berkeley, California (1987).

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

Pruess, K., Oldenburg, C. and Moridis, G.: TOUGH2 user's guide, version 2.0. LBNL-43134, Lawrence Berkeley National Laboratory, Berkeley, California (1999).

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).

Saad, Y. and Schultz, M.H.: GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.* **7(3)**, 856 - 869 (1986).

Sleijpen, G.L.G. and D. Fokkema: BiCGSTAB(*l*) for linear equations involving unsymmetric matrices with complex spectrum, *Electronic Transactions on Numerical Analysis* **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).

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).

Zyvoloski, G.A., Robinson, B.A. and Viswanathan, H.S.: Generalized dual porosity: a numerical method for representing spatially variable sub-grid scale processes. *Adv. Water Res.* **31**, 535 – 544 (2008).