Proceedings World Geothermal Congress 2015
Melbourne, Australia, 19-25 April 2015

OOMPFS – A New Software Package for Geothermal Reservoir Simulation

Peter Franz

Mighty River Power, 283 Vaughan Road, PO Box 245, Rotorua 3040, New Zealand

Peter.Franz@mightyriver.co.nz

Keywords: Two Phase, Heat, Flow, Simulator, OOMPFS

ABSTRACT

The demand from a geothermal modeler's point of view has grown beyond the features provided by existing geothermal reservoir simulators. Existing packages are either built on code which is over 30 years old (e.g. TOUGH2) or is proprietary (e.g. TETRAD); in both cases adapting the code to include new features required for simulating a developed geothermal field is challenging. In particular, the representation of wells is crude, and surface networks and plant models do not exist or are so limited that they restrict the capability to predict output of geothermal power plants in long-term scenarios.

OOMPFS is a new Object Oriented Multi Phase Flow Simulator designed specifically for geothermal reservoir simulations. Its internal computational core has been written in C++ and is loosely-based on the well-proven methodology of TOUGH2 for solving the non-linear problems encountered in geothermal simulations. However the object oriented, open structure has made it possible to rapidly add the functionality which was lacking in existing reservoir simulators. Using modern programming techniques and freely available software libraries make the code and simulations very flexible. A simple graphical user interface facilitates the creation of new simulations and eliminates a large source of user errors. For very complex special cases the C++ libraries can be used similarly to PyTOUGH. Model grids are established in 3D using the Visualization Toolkit (VTK) library and thus simplify many of the pre- and post-processing steps. Equation of State (EOS) classes include an unlimited number of tracers; custom classes with different thermodynamic properties can be easily implemented. Currently the most important EOS classes using air, CO2 and brine are included in the package. Wells are represented realistically, including coupling to wellbore simulators, surface networks and plant models. Custom smart plant models allow treatment of scenarios as realistically as desired by the user. Aside from standard reporting tools, the user can write custom reporters that can report any desired entity available through the simulation. Parameter estimation, sensitivity analysis and uncertainty propagation can be easily performed by using the iTOUGH2-PEST interface.

1. INTRODUCTION

Numerical models play a key part in the development and operation of a geothermal field. Reservoir simulators like TOUGH2 and TETRAD are used for solving the non-linear equations governing the transport of mass and heat. A particular challenge in the geothermal environment is the need for these simulators to simulate non-isothermal conditions including phase changes.

Unlike the oil and gas industry, the development of new reservoir simulators in geothermal has been very slow. The de-facto industry standard tool is TOUGH2, which is built on a base of over 30 year old FORTRAN code, with the current version 2.0 dating to 1999. While the TOUGH2 engine is an admirable simulator even today, its lack of adaptability to modern programming techniques has precluded further development on the code.

Apart from user-friendliness – which is a major hurdle for TOUGH2 users – TOUGH2 lacks important features which are desperately needed by today's geothermal modelers. The most striking deficiencies are the lack of a proper wellbore simulator and a surface network model. Only with these two features tightly coupled into a simulation package is it possible to properly calibrate a model to observed changes in wellhead flow, pressure and enthalpy, and to perform uncertainty propagation on well and plant behavior in future scenarios. These capabilities are of high interest to operators and owners of geothermal fields who need to plan and finance the future operation of the field.

TOUGH2 was mainly developed before object oriented programming (OOP) become the main stream coding paradigm in the mid 1990's. In OOP the code is divided into many small "objects" which contain data and methods to work on this data; the encapsulation of the objects facilitates their reusability and interchangeability. New features can be rapidly coded and interchanged, and large libraries are accessible for different complex tasks so the programmer can efficiently make use of code written by others.

OOMPFS (Object Oriented Multi Phase Flow Simulator) started as a small test project in 2011 using MATLAB to see how difficult it would be to set up a software framework to solve two-phase Darcy flow in a geothermal reservoir. Encouraged by quick success a C++ version was started in mid-2013 which quickly grew into a full simulator. Emphasis was taken to include as much open-source code as possible to avoid unnecessary duplication of work. Despite some setbacks the code quickly developed into a working, stable simulator. Work on OOMPFS is ongoing, but it already accomplishes the main development goals including wellbore and power plant models. The desired uncertainty propagation, plus sensitivity analysis and parameter estimation, are handled using a very easy-to-use coupling to iTOUGH2-PEST.

This paper serves as an introduction to the capabilities of the OOMPFS software package. To show all its features and validation in detail is beyond the scope of this work; focus is rather given on its key features and some explanatory examples.

2. THE RESERVOIR SIMULATOR

The reservoir simulator part of OOMPFS is based on the computational concepts introduced in TOUGH2 (Pruess et al., 1999). Most of the equations governing the non-isothermal Darcy flow are the same or very similar to the ones given in the TOUGH2 manual.

The mass and energy inside an element at a future time is related to the mass and energy present at current time considering sources/sinks and transport into or out of the element. The residuals for mass/energy are calculated in the form

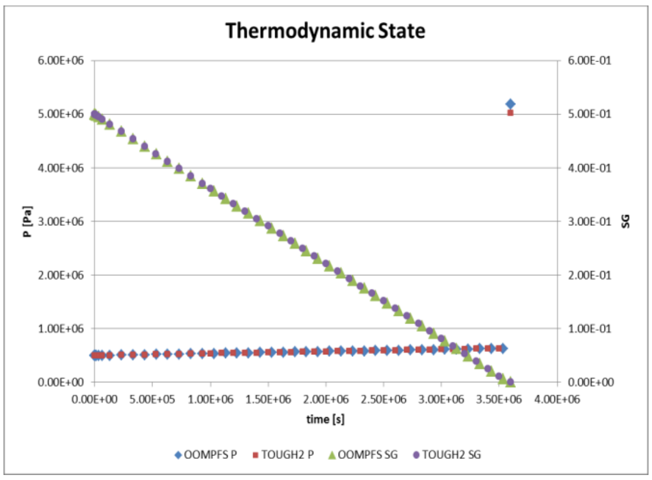
$$R(t+\Delta t) = M(t+\Delta t) - M(t) - \Delta t * (Q - S - F)$$

where R describes the residual, M the mass or energy present, Q a source rate, S a sink rate and F the Darcy-Flux between elements. M(t) is a known quantity; M(t+Δt) is computed from the primary variables and Q, S and F depend on the primary variables. Hence R becomes a function of the primary variables; the Newton-Raphson method is used to determine the primary variables through an iterative process. The main computational task is building the Jacobian matrix needed for the iterative method by using numeric differentiation. The iterative process is stopped once convergence criteria have been achieved.

2.1. Equation of State

The equation of state in OOMPFS is fully built in the object-oriented paradigm. Its encapsulated form enables OOMPFS to work with any set of primary variables which describes the multi-phase flow problem adequately. However it was found that some primary variables perform better than others within the Newton-Raphson solver. For example, while using (pressure, enthalpy) as primary variables it was found that despite the advantage of having a set of consistent primary variables the slower convergence in the Newton-Raphson solver was a major drawback. Most of the focus has since been given to use the "traditional" (pressure, temperature/saturation) formulation as used in TOUGH2. However, it is planned to investigate the potentially very benign (density, internal energy) formulation, which should combine the advantages of having a consistent set of primary variables with fast convergence in the Newton-Raphson solver. Before this can happen an adequate steam table needs to be developed (Pruess et al., 1979).

To test the different equation of state classes it was found most useful to generate small, one-block models with either a source or a sink attached. These small test models give insight into the current thermodynamic state; also by monitoring the source/sink terms conservation of mass and energy can be confirmed. Figure 1 shows the results of such a test model for pure H_2O. A source of W=0.1kg/s, H=1200kJ/kg is attached to a block with V=8000m3, φ=0.1 and with initial state p=50bar, SG=0.5. The thermodynamic state is compared to the same model setup in TOUGH2; the agreement is excellent. The gas saturation is decreasing until phase change occurs, at which state the pressure increases rapidly. Note that the small difference in the pressure at ~51bar is only due to a difference in time-step between OOMPFS and TOUGH2 – a common problem which often makes direct data comparison difficult. The right hand side plot in Figure 1 shows the mass and energy present in the block; both increase in accordance with the injected mass rate and enthalpy.



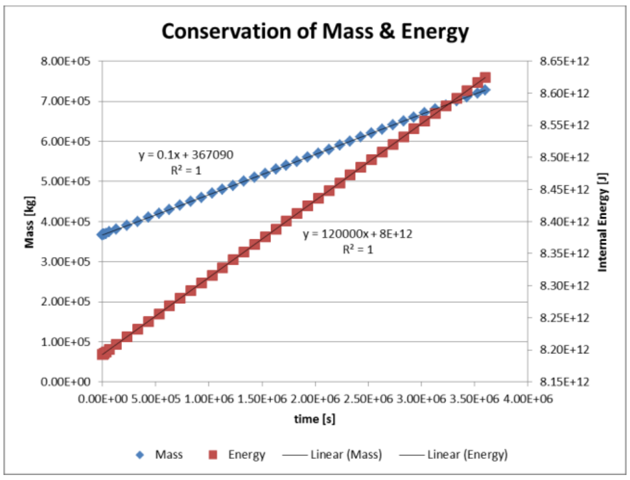


Figure 1: Left: Thermodynamic state of the single block model in comparison to TOUGH2. Note the slight discrepancy at P=5e6Pa is due to a slight difference in plotting times. Right: Conservation of mass and energy in OOMPFS.

A number of different EOS classes have been set up in OOMPFS and tested in the above fashion. In particular, EOS for water/air, water/CO_2, water/air/brine and water/CO_2/brine have been implemented.

2.2. Flow between Elements

Similar to the one-block model set up for testing the thermodynamic state of a particular EOS, simple two-block models can be used to investigate the flow through a common boundary. Of particular interest is the influence of gravity, i.e. if the blocks are side-by-side or stacked on top of each other. However at this stage validation becomes difficult; it is hard to determine if the resulting transients are due to thermodynamic issues or due to transport through the common interface. Nevertheless a number of manual calculations have been done to validate the flow between blocks.

Figure 2 shows the result of such a two-block model, using two blocks stacked on top of each other. The two blocks are initialized with different pressures and gas saturation. Two-phase flow results in a declining pressure in the upper block and increasing pressure in the lower block. Essentially what can be seen in the model is interchange of liquid downwards due to gravity. At around t=8.1*10^{5}s the upper block has lost all its liquid phase; the remaining steam rapidly equilibrates the pressure. Comparison between OOMPFS and TOUGH2 shows excellent agreement for this model.

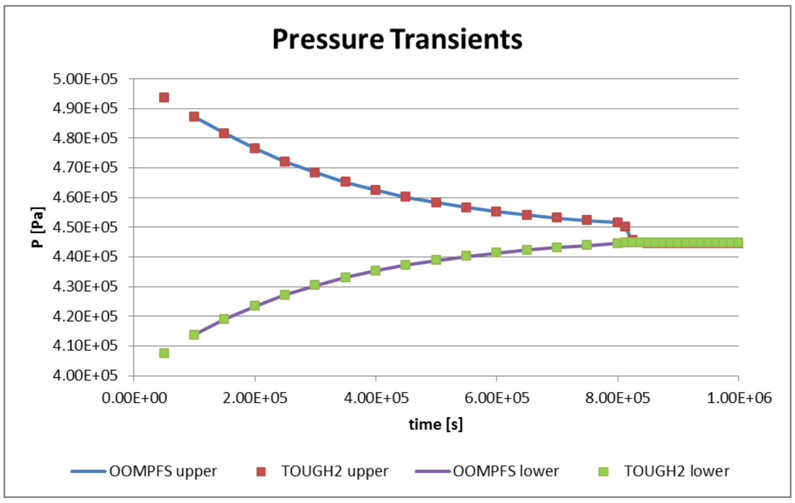


Figure 2: Pressure transient between two vertically stacked blocks.

2.3. Five-Spot Geothermal Model

The TOUGH2 manual describes a simple five-spot geothermal model (problem number 4) which was set up in OOMPFS and TOUGH2 for inter-comparison. This model contains an injection well and a producing well working at the same mass flow rate (7.5kg/s). The model describes the transient of the cool injection fluid flowing towards the producing well.

The geometry described in the manual was slightly altered since PetraSim was used as a pre-processor. PetraSim does not allow for the grid block to be set up as described in the manual. Instead a rectangular 10x10 grid with a block width of 50m was used. All other parameters are as described in the TOUGH2 manual.

The identical model was set up in OOMPFS. Both the TOUGH2 and the OOMPFS model were run for 36.5years simulation time. The model was run three times: once as a single porosity model and twice as a double porosity model using 50m and 250m fracture spacing. The results are shown in Figure 3; for all three models the results are virtually identical.

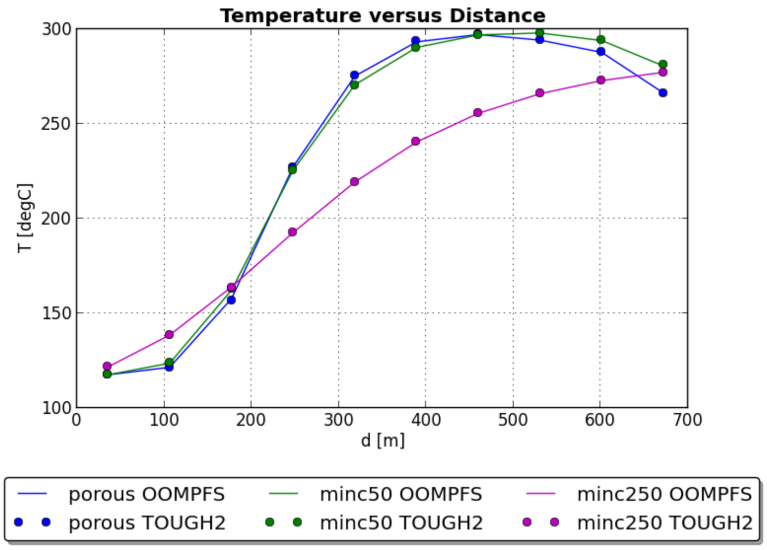


Figure 3: Five-Spot Geothermal Model (TOUGH2 problem number 4). Shown is temperature versus distance from the injector. The results from OOMPFS are shown as solid lines; the corresponding TOUGH2 results are shown as solid circles.

2.4. Larger Models

Several TOUGH2 models internally used at MRP were transferred to OOMPFS using Python scripts. The comparison of the natural state runs of TOUGH2 and OOMPFS always yielded very similar though not identical results. Natural state pressure and temperature match very well between the two simulators, but gas saturation quite often displays small differences. These are thought to be mainly due to different weighting of phase densities at the block interfaces: whereas TOUGH2 uses the mean of the two densities OOMPFS uses a weighted average with the connection distances as weighting factors. This results in small differences when the grid contains layers with uneven layer widths.

The largest model translated from TOUGH2 and run in OOMPFS so far contains about 18,000 elements (double porosity, i.e. 9000 blocks). It has provided stable runs in OOMPFS using wellbore simulation and the surface network; however the overall runtime for the calibration period is about 4-5 times longer than the TOUGH2 version.

3. COUPLING TO WELLBORE SIMULATOR

One of the key development goals of OOMPFS was to couple the reservoir model to a geothermal wellbore simulator in order to use the package for realistic production scenarios. Per se this coupling is not very difficult. The reservoir simulator provides the wellbore simulator with thermodynamic conditions at the feedzones of the well. The wellbore simulator then calculates the flowing conditions to achieve a certain mass flow at the wellhead and then passes the feedzone mass allocations back to the reservoir simulator.

The main difficulty – apart from writing the code for the coupling – is to find a wellbore simulator which is robust enough to use for the coupling. Many of the wellbore simulators initially tested were not able to generate high quality or stable output under adverse conditions at the feedzones; particularly wells with multiple feedzones are very hard to model.

The coupled wellbore simulator used for this work is GeoWell (purchased from Mauro Parini, Ph.D. Consultant, Geothermal Reservoir Engineering); however it is planned to replace GeoWell in OOMPFS with a purpose-built code (PAIWERA) that is in process at the time of writing.

During calibration and production scenarios each well is wellbore simulated separately with a deliverability curve of mass flow rate versus well head pressure, either at every time-step (computationally expensive) or at user-defined intervals. After selecting the desired flow rate the corresponding wellhead pressure is determined and a detailed wellbore simulation is run. This wellbore simulation determines the mass rates for the feedzones for the next time-step(s). Results for each deliverability curve and wellbore simulation can be saved and analyzed over time.

Care is taken to update flowing enthalpy and mass components at every time-step and also for determining the Jacobian matrix; failure to do so would quickly lead to failure conditions in the reservoir simulator. Updating the mass flow rates at every time-step for use in the Jacobian matrix was found too computationally expensive; also using the wellhead pressure search functionality in the wellbore simulator makes the simulation not really useful for numerical differentiation. Fully updating the feedzone mass flow rates at user-defined intervals has proven to be the best compromise, although in tight reservoir permeability/porosity conditions it is easy to completely drain all fluid from a block, resulting in reservoir simulator failure. When such a condition occurs it is best to repeat the simulation using a higher update frequency.

To demonstrate the wellbore coupling and the surface model a small 10x10x3 grid block model was generated. Four production wells feed a power plant with subsequent 50% shallow reinjection. The model is run in the coupled mode for 30 years to demonstrate the change in characteristics. Figure 4 shows a snapshot in time of the model.

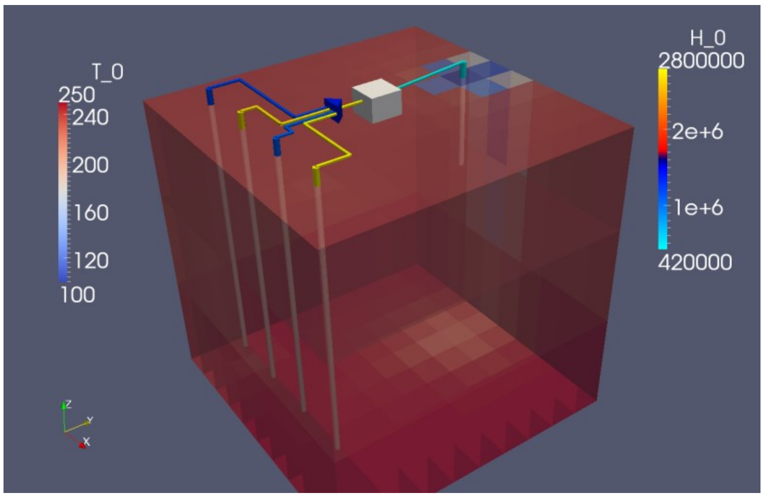


Figure 4: The demonstration model. The reservoir is colored by temperature (°C); the surface pipelines are colored by enthalpy (J/kg). This snapshot was taken with Paraview, which can directly display the output files of OOMPFS without further conversions needed.

Figure 5 shows the deliverability curves of Well 1 of the model over time. The maximum flow rate of the well is decreasing significantly over time and the enthalpy is increasing, particularly in the later stages of the simulation. This has implications for the plant, which will be discussed below.

Figure 6 shows a 3D representation of the wells colored by their mass flow rate. Note that two of the wells have their main feedzone in the shallower regions; in this case the mass flow below the feedzone is zero. Virtually all data, like steam quality, velocities, temperature, etc. from the borehole simulation is stored and can be either used for further numeric analysis or for visualization in 3D.

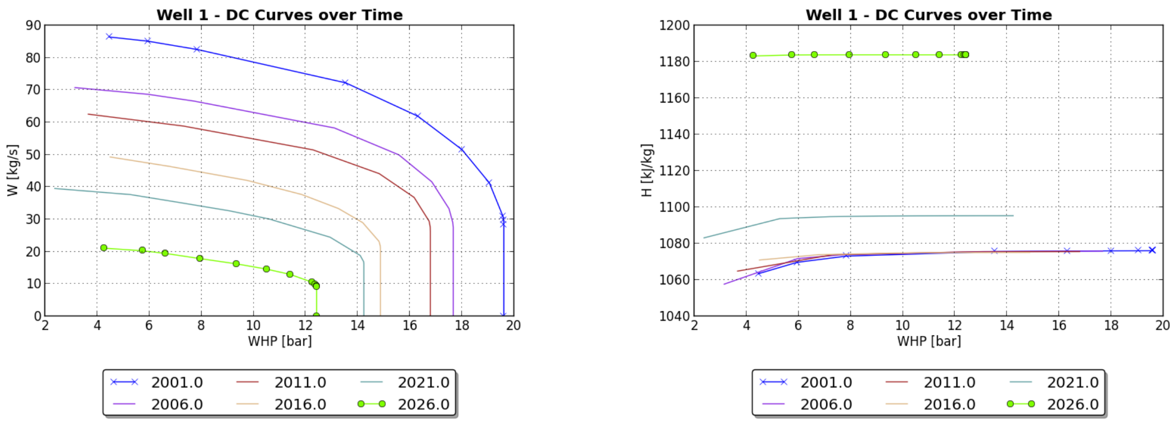


Figure 5: Deliverability curves (DC) for mass flow (left) and enthalpy (right) at the wellhead of the demonstration model.

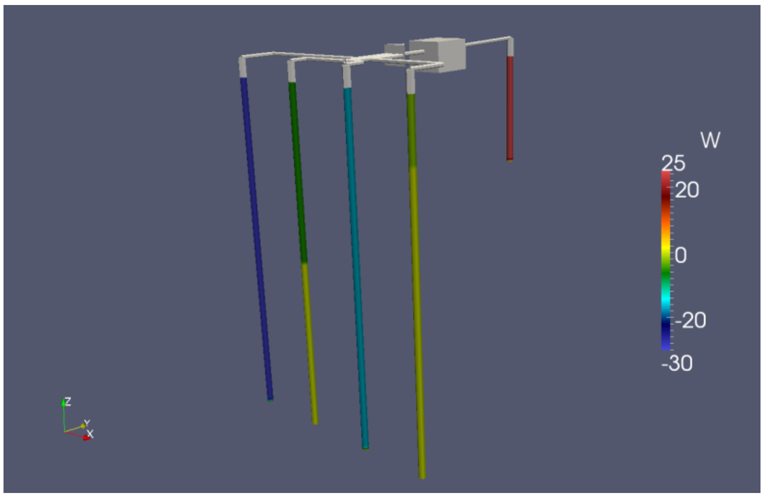


Figure 6: The producing wells and the injector in the demonstration model. The wells are colored by mass flow rate, where negative values indicate production and positive values indicate injection.

4. THE SURFACE MODEL

The surface model in OOMPFS aims at simulating all activities at the surface in enough detail to allow for realistic production scenarios. It simulates the flow coming out of the producing wells, conversion to electricity and subsequent reinjection into the reservoir. The fundamental principle of the surface network is the conservation of mass and energy; detailed simulation of pressure (e.g. in surface pipes) is not yet implemented but may be added in the future.

All surface objects are described in the object-oriented method using a common interface. Hence the objects become fully interchangeable; all the user needs to do is to create an object, then define input and output connections and auxiliary parameters (like a pressure for a separator object).

Certain objects, for example power plants, are *smart* objects which can interact with the surface network to optimize targets. Internally they use a Levenberg-Marquardt algorithm to minimize their objective function; for example a steam plant can adjust the flow rate of steam to generate a fixed amount of power. Achieving more than one target is also possible, for example generating a set power rate while simultaneously minimizing the amount of fluid extracted by making optimum use of the available wells.

Makeup wells, set up initially at zero flow, can be brought online once the generation target cannot be achieved. Also wells that cannot generate any more (i.e. maximum discharge pressure is too low) are automatically abandoned.

Results from the demonstration model are shown in Figure 7 and Figure 8. The plant has a set target of 60MW of thermal generation. Initially the flow from Well 1 and Well 2 is sufficient to provide this target. In 2008 Well 3 needs to be brought online, followed by Well 4 in 2009. With these additional wells the production target can be met until 2015, whereon generation declines due to the lack of additional makeup wells. Note that the Levenberg-Marquardt algorithm achieves a very stable generation target over a period of significant change in the production mix.

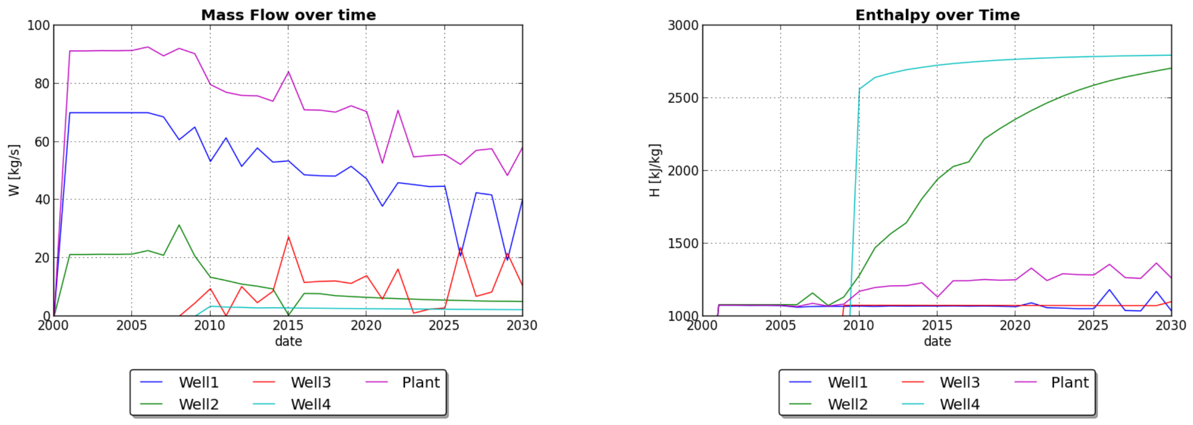


Figure 7: Mass flow rates and enthalpy of the producing wells and the plant over time.

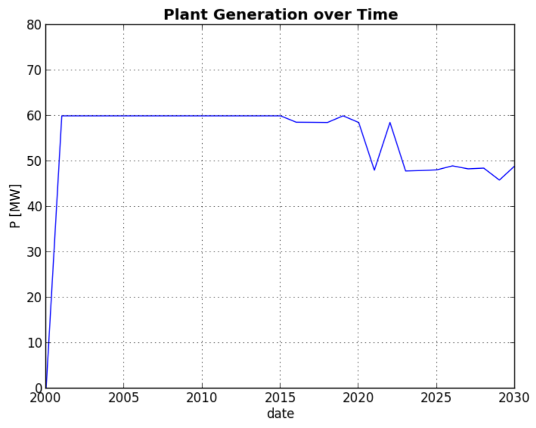


Figure 8: Thermal generation rate of the plant in the demonstration model.

5. INTEGRATION OF ITOUGH2-PEST

Sensitivity analysis, parameter estimation and uncertainty propagation are implemented in OOMPFS via a coupling to iTOUGH2-PEST (Finsterle, 2007 and Finsterle, 2011). The PEST interface allows virtually any simulator that uses ASCII based input files to use the capabilities of iTOUGH. All input files to OOMPFS are either ASCII/XML text based or can be created or modified via small Python scripts; this allows for a very easy coupling to iTOUGH2PEST. Further, the graphical user interface in OOMPFS assists the user with setting up iTOUGH2PEST input files in the correct format, and additional auxiliary Python scripts have been developed to convert OOMPFS output to a format readable by iTOUGH2-PEST.

To demonstrate the coupling to iTOUGH2-PEST the demonstration model was used to investigate the uncertainty in the timing of the makeup Wells 3 and 4. The uncertain parameter chosen was the reservoir porosity; a normal distribution with mean 0.1 and standard deviation 0.01 was used. Eleven Monte-Carlo runs were completed and the time for bringing Well 3 and Well 4 online was used as an observation.

The results are shown in Figure 9. Well 3 is brought online at a mean time of 2007.9 and standard deviation 0.8. Well 4 is always brought online at either the same time or at a later stage; the mean time to bring it online is 2008.7 with a standard deviation of 1.0.

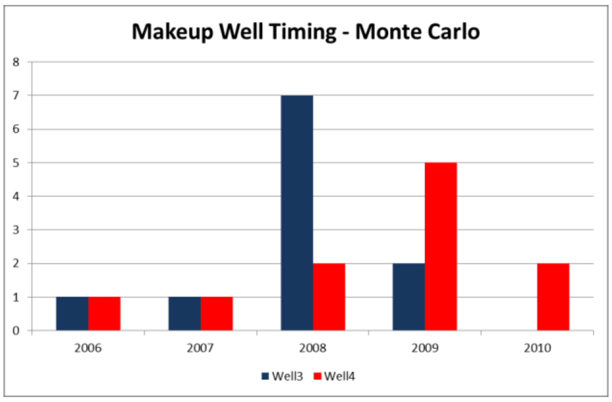


Figure 9: Makeup well timing from Monte Carlo simulation, binned by year in which the well is brought online.

6. ISSUES AND OUTLOOK

At the time of writing OOMPFS is providing very stable runs; typically the simulation only halts when an element moves outside the boundaries specified by the steam table, when a set minimum time-step is reached or, most frequently, when the wellbore simulator fails to find a solution. Therefore currently the main focus is on developing PAIWERA, a wellbore simulator that is robust in fully-coupled mode. The main challenge for the wellbore simulator arises when multiple feeds make the bottom to top integration along the wellbore difficult due to an a-priori unknown bottom hole pressure domain. PAIWERA shows promise in resolving this issue.

A very important issue is the computational time per time-step. Depending on the simulation, OOMPFS currently requires about 5-10 times more computational time than a similar model set up in TOUGH2. The reason for this is well understood – TOUGH2 is a monolithic, bare-thread code which is very fast but very inflexible. OOMPFS on the other side is very flexible but comes at the price of computational speed. The main target for OOMPFS is to run normal sized models (up to around 100,000 elements) on a standard desktop machine. To make this practical the computational time needs to be reduced. Since the main computational cost is currently in the determination of the Jacobian matrix it is possible to reduce overall computational time by parallelizing this task. This is not easy but achievable.

The task is finding a more suitable iterative solver. Several solvers have been tested so far; they displayed a wide difference in speed and maximum problem size. Some have displayed unforeseeable behavior – one of the solvers tested showed good computational speed and the ability to resolve large problems, but showed non-deterministic behavior, probably through an internal random number generator which rendered it useless for sensitive tasks.

3D representation of the output is much advanced but the input, i.e. generation and initialization of a grid from a conceptual model, is still cumbersome. PetraSim has been used as a 3D editing tool, with subsequent transfer to OOMPFS format using PyTough and other Python scripts. This is all right for the small models tested so far but will become harder for larger models in the future.

Validation remains a very hard task. With very few analytical solutions to two-phase problems available in published media, the main focus of validation has so far been reduced to comparing similar models between OOMPFS and TOUGH2. The geothermal community would greatly benefit from collating and publishing reference models which could be used for simulator validation.

REFERENCES

Pruess, K., Oldenburg, C., & Moridis, G. (1999). TOUGH2 User's Guide, Version 2.0. *LBNL-43134*.

Pruess, K., Schroeder, R. C., Witherspoon, P. A., & Zerzan, J. M. (1979). SHAFT78, A Two-Phase Multidimensional Computer Program for Geothermal Reservoir Simulation. *LBL-8264*.

Finsterle, S. (2007). iTOUGH2 User's Guide. *LBNL-40040*.

Finsterle, S. (2011). iTOUGH2 Universal Optimization Using the PEST Protocol User's Guide. *LBNL-3698E*.

Finsterle, S. (2012). iTOUGH2 Command Reference. *LBNL-40041*.