PROCEEDINGS, 42nd Workshop on Geothermal Reservoir Engineering
Stanford University, Stanford, California, February 13-15, 2017
SGP-TR-212

Algorithm for Optimal Well Placement in Geothermal Systems Based on TOUGH2 models

Dagur Helgason, Ágúst Valfells, Egill Júlíusson

Reykjavík University, Menntavegi 1, 101 Reykjavík, Iceland

dagurinn@gmail.com

Keywords: resource management, well placement, algorithm, optimization, tough2, python, pytough, reservoir engineering

ABSTRACT

In a world of ever increasing use of renewables geothermal has lagged behind and has seen little growth compared to other renewables due in part to its high capital cost. Geothermal wells account for about a third of the capital cost and it is therefore important to ensure the highest possible success rate and value creation from these wells. In order to address this, an algorithm has been developed that utilizes a numerical TOUGH2 model of a geothermal system to evaluate the optimal well placement based on a net present value estimation. The algorithm does this by using forward simulation of production for multiple potential well locations and estimating the net present value of each potential location. The algorithm is capable of using both deliverability wells and wellbore files. The algorithm was tested using a hypothetical model and found the optimal wells to be in the hottest parts of the model at depth, when using the deliverability setup, and in the upper heat zone, directly above the heat source, when using the wellbore file setup. The algorithm shows promise but has some faults and limitations, especially in the deliverability setup.

1. INTRODUCTION

Renewable energy in the world is growing at a faster pace than before, with an average growth rate of 12% across all sources (geothermal, hydro-power, solar PV, CSP and wind) in 2015. Geothermal, however, is growing slower than other sources with an average growth rate of 2.4% in 2015 REN21 (2016). This slow growth is affected by a variety of factors, one of which is the capital intensity of new geothermal operations. The U.S. EIA (2016) estimated a cost of $6,230 per kW of installed capacity for geothermal compared to, for example, onshore wind at 2,213/kW and large scale photovoltaic at 3,873/kW. The majority of this high capital cost comes from building the power plant and creating the wells, in Iceland the construction of the power plant accounts for approximately 35% of the capital cost and the drilling and well construction account for approximately 34% (Gehriner and Loksha (2012)).

With this high cost of well construction, it is important that as many wells as possible are successful and that they be as profitable as possible. The IFC (2013) estimates that the success rate of the first well in a new field is approximate 50% and that the cumulative success rate of the first 30 wells is approximately 70%. The success rate of the exploration phase (first 6 wells) in geothermal development has been steadily rising over the past decades but success rate of wells drilled during the development (wells nr.7-30) and operation phase (all wells after the first 30) has not increased as significantly IFC (2013). If the success rate of wells can be increased further the results would be a decreased capital cost as fewer wells would need to be drilled and increased profitability, which would make investors more easily attracted.

As an attempt to increase the success rate and profitability of geothermal wells an optimization algorithm (hereby referred to as the Algorithm) has been written that utilizes TOUGH2 Pruess, Oldenburg and Moridis (1999) forward modeling of production to estimate the net present value (NPV) of multiple potential wells and uses these estimations to determine the optimal well placement within the model. The project was done as a part of the research conducted by the Operations Research and Subsurface Modeling (ORSM) group which is a collaboration of Reykjavík University, the University of Iceland and the National Power Company of Iceland. The inspiration for the project was the work done by Basil Jefferies (2016) who used TOUGH2 forward modeling of production to simulate multiple potential well locations in order to determine the best location. Jefferies work was specific to Þeistareykir in the North of Iceland and the results of each simulation were input into excel where the NPV was then calculated. This project seeks to automate a similar process that should be capable of using most TOUGH2 models to determine an optimal well location.

2. METHODS

The Algorithm presented here uses the TOUGH2 numerical reservoir simulator along with Python (Lutz (2013)) and PyTOUGH (Croucher (2015)) to simulate possible well locations and determine the most profitable well location based on NPV evaluation. The algorithm was tested on a hypothetical model in order to evaluate its effectiveness.

2.1 Algorithm Structure

The Algorithm reads a numerical reservoir model, that the user supplies, and then proceeds to generate all the needed input files to simulate the possible wells. A possible well in this case is a cell in the model, the Algorithm generates one input file for every cell in the model each one with a single well added to a single cell. Once these files have been created the Algorithm proceeds to simulate all of

them creating the output files that are then used to evaluate the NPV of each possible well and arrange them into a list in order of NPV with the optimal well, defined as the well with the highest estimated NPV, at the top.

The Algorithm is split up into three main parts (PreProcessor, Processor and PostProcessor) as well as an algorithm (Generator) that generates scripts with the appropriate values for each run as well as making sure that the algorithms run in the correct order. The Algorithm uses base algorithms for the PreProcessor, Processor and PostProcessor, the Generator opens these scripts, creates copies of them and adds the relevant variables to each one. A flow diagram that shows the order of operations can be seen on Figure 1.

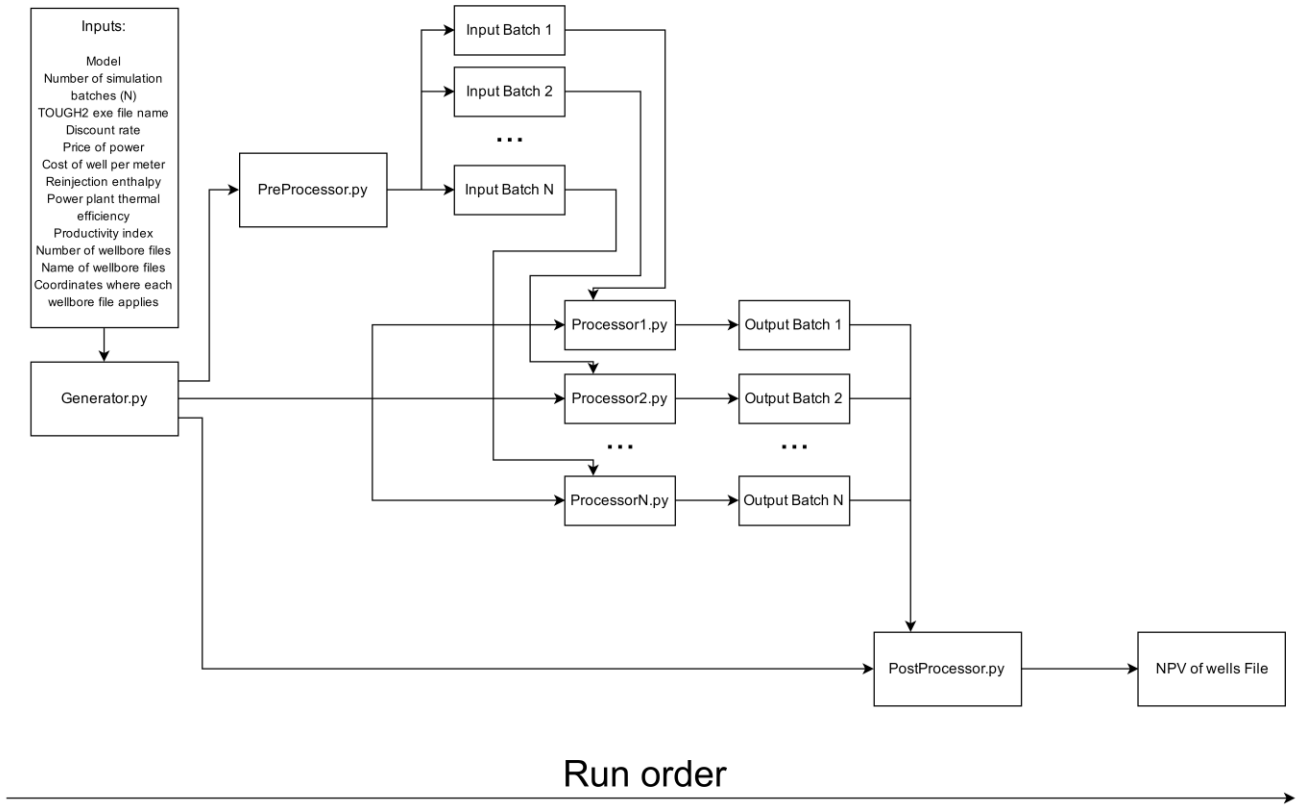

Figure 1: Flowchart describing how the Algorithm runs. The further to the right an operation is the later it is run. The user should run Generator.py which generates a PreProcessor.py script, a user defined number (N) of ProcessorXX.py scripts, a PostProcessor.py script and a batch file. The batch file makes sure the scripts are run in the correct order, first the PreProcessor.py, followed by all the ProcessorXX.py scripts and the PostProcessor.py script. PreProcessor.py creates the Input files needed for the simulations, these are sorted into N Input batches that are used as inputs for each of the ProcessorXX.py scripts. The Processor.py scripts each go into a simulation batch and run every Input file there creating N Output batches that contain the well data needed for the final step. The PostProcessor.py goes through the output files, evaluating the NPV of each well locations, and returns a file that lists all the simulated wells in order of their NPV, displaying their NPV, name and location coordinates within the model.

2.1.1 Well types

The algorithm uses two different types of production wells for its simulations, these are wells on deliverability and wells using wellbore files. The conditions where each type is used is discussed in section 2.1.4.

A well on deliverability uses an assigned bottom hole pressure in combination with the productivity index to calculate the flow in a well. A limitation of this is that the assigned bottom hole pressure will always be the same through the entire simulation. The algorithm also assigns the same bottom hole pressure to all wells which will skew the results in favor of higher pressure cells as the pressure difference between the bottom hole pressure and the cell pressure will be greatest there. It was decided to keep this functionality in the algorithm as it will still give a relatively good idea of where the best wells would be located even if the NPVs will be unreasonable at times.

A well on wellbore file uses an external file that defines the bottom hole pressure based on enthalpy and flow rate. These files are used to solve for flow rate and bottom hole pressure, based on the wells enthalpy, by iteration. This allows the bottom hole pressure to change with time resulting in a more realistic simulation. These files however have to be generated by a separate program, these programs are

called well simulators. A well simulator simulates the flow of a well for a variety of enthalpies and flow rates thereby calculating the bottom hole pressure. A limitation of this is that these files only have a limited range and if the well achieves conditions outside this range the simulation will simply stop. The wellbore file used in testing the algorithm is discussed in section 2.2.2.

2.1.2. Assumptions

In order to enable the Algorithm to run and estimate the NPV of the simulated wells certain assumptions had to be made concerning some of the parameters used. Many of the assumptions are made in order to simplify the subsequent calculations.

It is assumed that the TOUGH2 model is an accurate representation of the geothermal system as the models themselves do not show any uncertainty that can be used to show the confidence of the model or the variability from measured data. The price of power the operator gets is assumed to be known and that it is unchanged during the operation time. The cost of well construction is assumed to be known and that it is only dependent on the depth of the well, not being subject to the hardness of the rock, possible drilling and well construction complications. It is assumed that there are no operation and maintenance (O&M) costs associated with the well. Reinjection enthalpy is assumed to be the same for all possible well locations, this assumption is likely to skew the results as lower enthalpy wells would generally be able to reject a lower enthalpy than the higher enthalpy wells would.

2.1.3 Generator

This algorithm is used in order to generate the other parts of the Algorithm that then each run a specific part of the whole process. It takes the following inputs: model, number of simulation batches, TOUGH2 exe file name, discount rate, price of power, cost of well per meter, reinjection enthalpy, power plant thermal efficiency and the height/depth of surface, the remaining inputs are input into the PreProcessor.

Once the Generator has all the required inputs it creates a batch file called batch.bat and writes commands that run the PreProcessor, Processors and PostProcessor into it. The batch file is there only to run the algorithms in the correct order automatically, it will include one line for the PreProcessor, one line for each of the Processors and one line for the PostProcessor. The Generator then proceeds to open the base PreProcessor and make a copy of it called PreProcessor01.py, the base file has a placeholder that the Generator replaces with the following inputs in the copy: model and number of simulation batches. Similarly, the Generator will make a number of copies, equal to the number of simulation batches, of the base Processor, these files will be called Processor01py, Processor02.py and so on. These files will also have a placeholder that is replaced with the relevant inputs, in this case the following: model, number of simulation batches and TOUGH2 exe file name, the processor number (1 for Processor01.py, 2 for Processor02.py and so on) will also be added as an input here although it is not user defined. The Generator will then create a copy of the base PostProcessor called PostProcessor01.py, the base file includes a placeholder that is replaced by the following inputs: model, number of simulation batches, discount rate, price of power, cost of well per meter, reinjection enthalpy and power plant thermal efficiency. Finally, the Generator runs the batch.bat file thereby starting the generation and processing of data.

2.1.4 PreProcessor

The PreProcessor generates the TOUGH2 input files that will be simulated. It opens the model that is input into the Algorithm, adds a generator to the first cell in the model and saves that configuration as Input000001, it then removes the well and adds a generator to the second cell and saves that configuration as Input000002, it goes through all the cells in the model like this creating a file for each configuration.

Before the PreProcessor starts generating input files it asks the user to define the simulation length in years, the assumed productivity index, in m^3, as well as a user specified number of wellbore files. If the number of wellbore files is set to 0 all wells will be set up as wells on deliverability. If the number of wellbore files is larger than 0 the script will go through a short loop where the user specifies the name of the wellbore files and where in the model each applies. The PreProcessor uses this information to setup a series of lists that it can call on later to determine what type of well should be used in any given area. Next the PreProcessor defines the starting time as 0 and the end time is set to the user specified number of years that are to be simulated.

In order to keep the folder the Algorithm is running in (from here on referred to as the root folder) as tidy as possible the PreProcessor will create a folder called "temp" within the root folder. The PreProcessor will also create subfolders within the "temp" folder called "simulator1", "simulator2",… that will contain each of the simulation batches. The PreProcessor will move all the wellbore files that have been specified to each of the "simulator" folders. The PreProcessor will now proceed to generate the input files. It does this, like mentioned above, by opening the model that will be simulated, adding a well to the first cell, saving the configuration as a copy, removing the well and moving on to the next cell. Each time the PreProcessor places a well it starts by finding the center of the cell it is adding the well to, if this cell is within an area defined as an area where a wellbore file applies the PreProcessor will name the well type in the appropriate way so that TOUGH2 will call on the correct wellbore file during simulation. All the generated input files are added into the "temp" folder.

2.1.5 Processor

Once the PreProcessor has generated all the input files the Processors start running. They will each begin with moving the input files assigned to them into the appropriate "simulator" folder, Processor01 will move the files assigned to it into the "simulator1" folder, Processor02 will move the files assigned to it into the "simulator2" folder and so on. The Processors will also copy the TOUGH2 executable file from the root folder to each of the "simulator" folders, this is done to allow for parallel processing as trying to run the

same executable file for multiple simulations simultaneously will result in an error. Before starting its simulations, the Processors will also copy all wellbore files into each of the "simulator" folders.

Once all the relevant files have been copied to the correct folders the Processors will start simulating the input files assigned to them. This will then generate output files called Input000001.listing for the first simulation, Input000002.listing for the second simulation and so on.

2.1.6 PostProcessor

The PostProcessor will start running at the same time as the Processors will in order to save some time. It will start by going through each of the "simulator" folders and move all output files into the "temp" folder. If the PostProcessor attempts to move an output file that is still being generated it will run into an error and stop, to prevent this the PostProcessor starts by checking if the next file in sequence exists. By doing this it is ensured that the files being copied are not being used by the Processors and that a ready output file is what is copied. Once all the output files have been copied into the "temp" folder the PostProcessor will start reading them and evaluate the NPV of each well location. It starts with Input000001.listing then goes to Input000002.listing and so on.

The NPV of each well location is calculated using Equation 1 where n is the number of output times, t_i is the simulation time at which output i is printed in years, Revenues_i are the revenues from time t_{i-1} to time t_i, Costs_i are the costs from time t_{i-1} to time t_i and r is the discount rate.

$$NPV = \sum_{i=0}^{n} \frac{\text{Revenues}_i - \text{Costs}_i}{(1+r)^{t_i}} \tag{1}$$

For the sake of simplicity, the costs are assumed to be 0 for all years except the first one where the only costs are assumed to be the cost of constructing the well. The cost of the well is assumed, as mentioned above, to only be dependent on the depth of the well. To determine the depth of the well the Algorithm finds the coordinates of the center of the cell that has the well, that is being simulated in each case, and subtracts that from the user defined surface depth coordinates. This depth is then simply multiplied by the cost of well per meter to give the cost of the well which will be input as Costs_0 in Equation 2.

In order to make a conservative estimate of the NPV the revenues are always calculated using the well conditions at the end of each time period. The revenues are calculated from the well flow and enthalpy using Equation 2.2 where $\dot{m}_i$ is the mass flow of geofluid from the well at time t_i, h_i is the enthalpy of the geofluid at time t_i, h_r is the enthalpy of rejected geofluid (or reinjected), η is the thermal efficiency of the power plant and Price is the price of power to the operator.

$$\text{Revenues}_i = (h_i - h_r) \cdot \dot{m} \cdot \eta \cdot (t_i - t_{i-1}) \cdot \text{Price} \tag{2}$$

As mentioned in section 2.1.2 the assumption that the rejected enthalpy is the same for all wells will result in skewed results for some wells as lower enthalpy wells would be able to reject lower enthalpy fluids thereby extraction proportionally more energy than the Algorithm calculates. Further the revenues can result in a negative number if the well enthalpy is lower than the rejected enthalpy.

Once the NPV has been calculated for an individual well the PostProcessor will save the NPV as well as the name of the well, its location, the number of the simulation file and the last simulated output time to a list, the NPV for all wells are stored in the same list. When the NPV for all wells has been calculated the list is arranged in order of NPV with the highest NPV well at the top, this list is then output as a text file.

2.2 Testing

As mentioned above the Algorithm was tested using a hypothetical model (hereby referred to as the Hypothetical model). The Hypothetical model as well as the parameters used during the testing simulations are described here.

2.2.1 Hypothetical model

The Hypothetical model is a fairly simple model with three rock types, a base rock, a cap rock and a reservoir rock. The model geometry is cubical with a width of 10,000m in both x and y directions and a depth of 2,500m with 3050 cells. The thermal and physical properties of each rock type can be seen in Table 1, only the permeability of the rock types varies in this model. A picture of the model showing the location of the various rock types as well as the location of two corner points and their coordinates can be seen on Figure 2

Table 1: Properties of the three rock types in the Hypothetical reservoir model used for testing the algorithm.

Rock type	Density [kg/m^3]	Porosity [-]	XY-Permeability [m^2]	Z-Permeability [m^2]	Wet heat conductivity [W/m*K]	Specific heat [J/kg*K]
Cap rock	2600.0	0.1	$1.0*10^{-15}$	$1.0*10^{-15}$	2.0	1000.0

Reservoir rock	2600.0	0.1	$1.0*10^{-14}$	$1.0*10^{-14}$	2.0	1000.0
Base rock	2600.0	0.1	$1.0*10^{-17}$	$1.0*10^{-17}$	2.0	1000.0

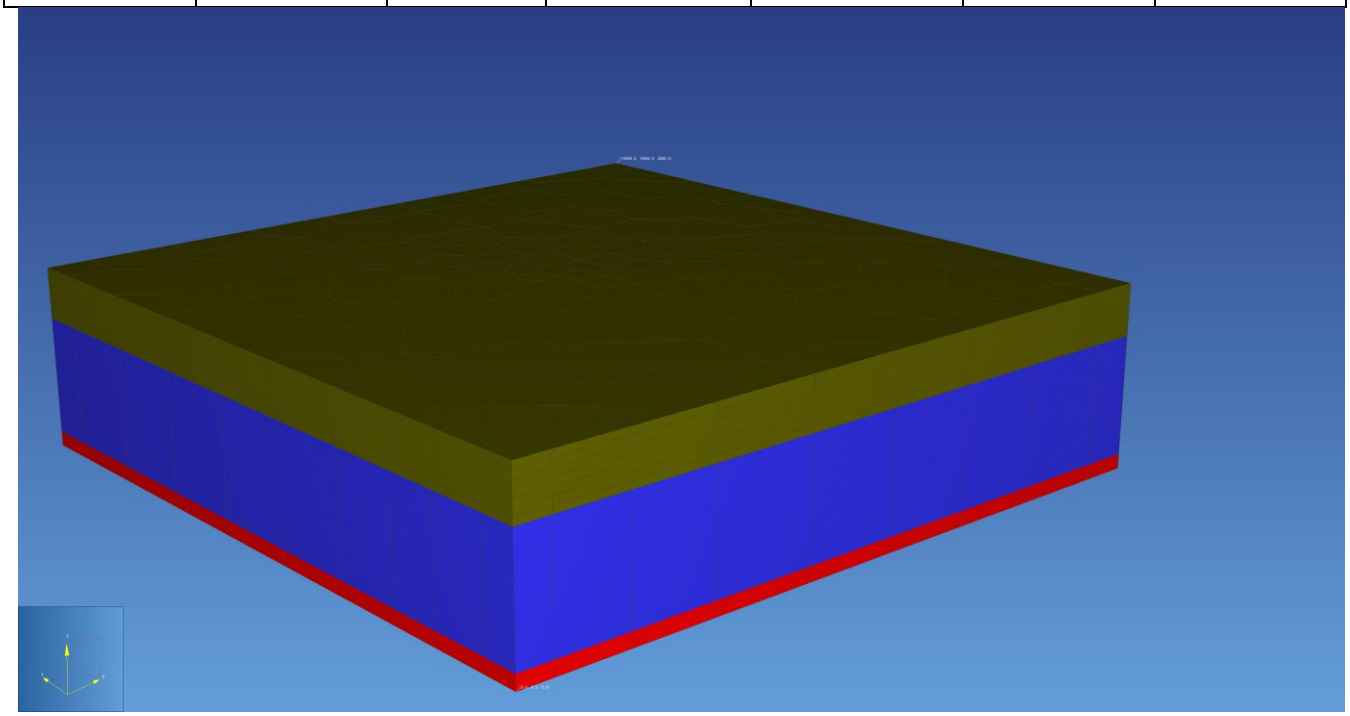

Figure 2: Visual representation of the Hypothetical model, used for testing the Algorithms, showing the three different rock types and their locations within the model. The top layers are the cap rock represented with a brown color, the reservoir rock is represented with a blue color and finally the bottom layers consist of the base rock represented with a red color. Two corner points are shown on the figure as well as their coordinates, these are at the bottom the point (0.0; 0.0; 0.0) and at the top the point (10,000.0; 10,000.0; 10,000.0)

The side, bottom and top boundaries are all set to fixed conditions. The side and top boundaries of the model are set to fixed conditions of pressure and temperature which are described by a temperature gradient and a pressure gradient. The surface temperature is set to 10°C and the temperature gradient is set so that the temperature at the bottom of the model is 100°C, this results in a temperature gradient of 36°C/km. The pressure at the surface is set to 100kPa, which is close to the average atmospheric pressure worldwide, and the pressure gradient is set to 7.4kPa/m which is about 25% below the hydrostatic pressure gradient.

The bottom layer follows these conditions but the center cells in that layer had different fixed conditions. The center cells are all cells with a center between x=3333m and x=6666m and between y=3333m and y=6666m. These cells had the same pressure as the ones at the outer boundary, as defined above, but had a fixed temperature of 450°C. These cells then act as a heat source for the system.

Once these conditions have been set the model is run till it reaches steady state. Figure 3 shows plane slices of the temperature in the model at steady state.

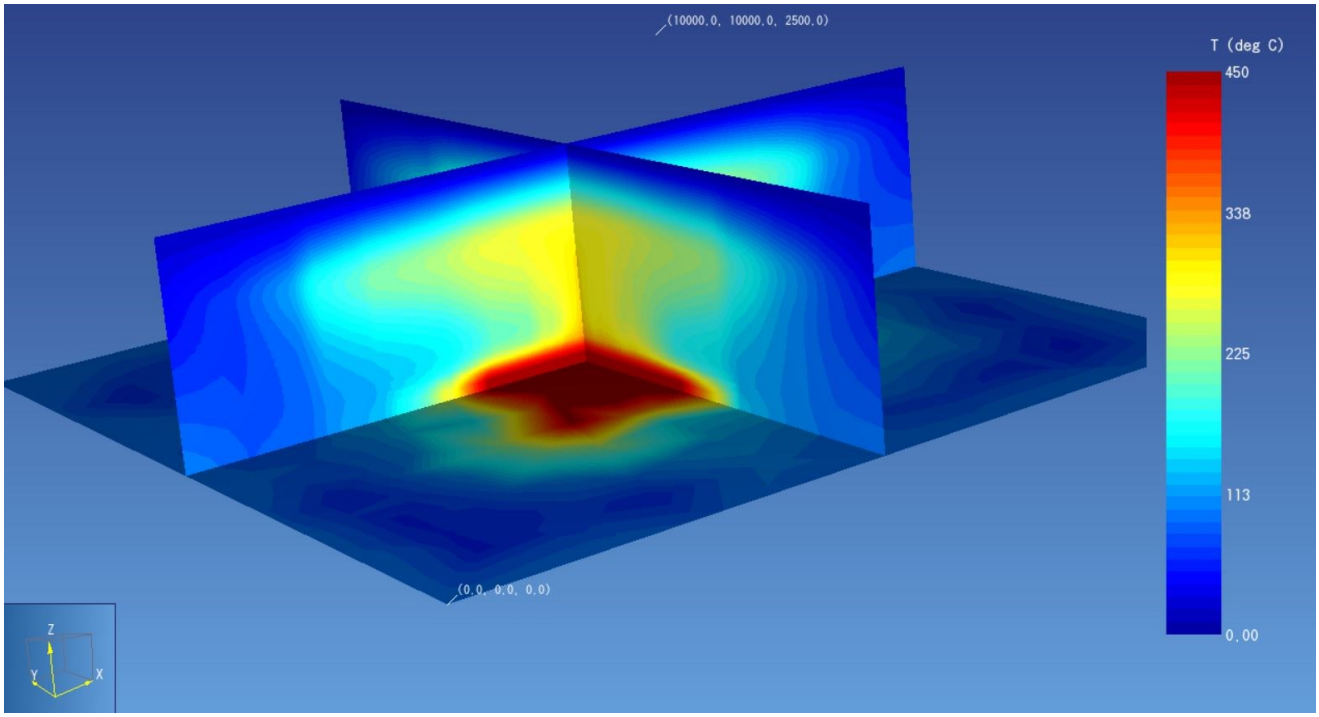

Figure 3: Plane slices showing the temperature of the Hypothetical model at steady state.

2.2.2 Running the algorithm

The Hypothetical model was run through the Algorithm with all wells set to well on deliverability on the one hand and with a wellbore file on the other. The wellbore file that was used is from well BJ-14 at Bjarnarflag in northern Iceland. This file was lent to the author by ISOR (Iceland GeoSurvey) for research purposes along with several other files including the wellbore files for BJ-11 and BJ-13. The file for BJ-14 was chosen for use on the basis that it had the largest enthalpy range of the available files. This file was made by Sigríður Sif Gylfadóttir in 2013 using wellbore simulation to mimic the output from the well, Figure 4 shows the simulated well bottom pressure of the well as a function of enthalpy and mass flow.

The Algorithm was run using the same values for numerical inputs in both the well on deliverability and the wellbore file case. The values used were the following

- A productivity index of $2.5*10^{-12}$ m^3 was chosen as it is within the simulated and calculated ranges at Bjarnarflag (Gylfadóttir (2013)).
- Discount rate is set to 0.07 (7%) as this is a discount rate that the National Power Company of Iceland would most probably use (Júlíusson (2016)).
- The price of power to the operator is set to $30/MWh, this value is based on the average LCOE of new geothermal plants as estimated by the EIA (2016). The value estimated by the EIA is $45.0/MWh as that accounts for all parts of the power plant this number needs to be lowered as the algorithm only accounts for the well itself. To do this the cost as estimated by the EIA was halved and rounded to the nearest whole $5 as this is a very rough estimate.
- The cost of the well is set to $2,000/m and is based on estimates from Kipsang (2013). Kipsang used a 3,000m sample well and estimated the cost of that well to be $6,045,000 or roughly $2,000/m.
- The rejected enthalpy is set to 1000kJ/kg. This value is chosen as a compromise between getting unreasonably high NPVs by setting the value too low and getting unreasonably few or no positive NPVs by setting it too high.
- The thermal efficiency is set to 0.1 (10%) and is chosen as a lower end efficiency based on results from Moon & Zarrouk (2012).
- The well bottom pressure for the deliverability case is set to 50 bar-a as a value that is included in lower mid-range of the wellbore file for BJ14 (Gylfadóttir (2013)).

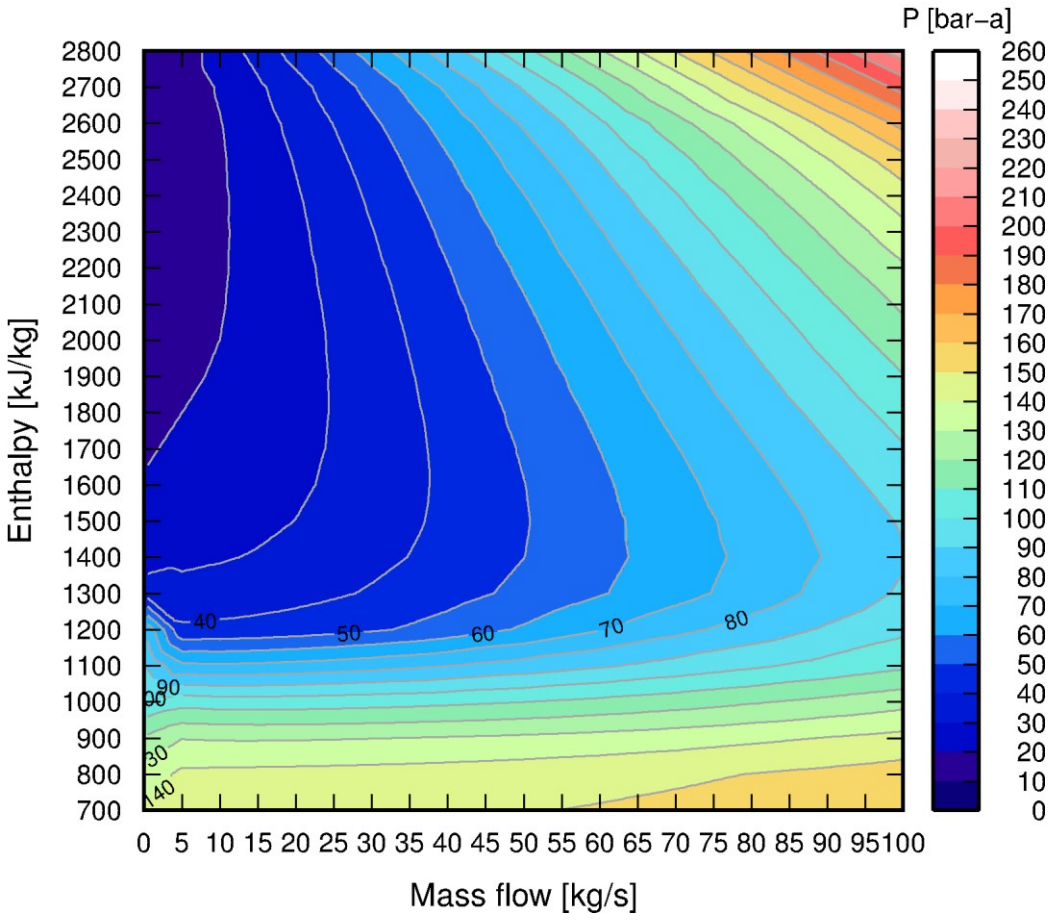

Figure 4: Well bottom pressure as a function of mass flow and enthalpy for well BJ14 in the Bjarnarflag geothermal field (Gylfadóttir (2013)).

3 RESULTS

The optimal well placement as discussed in chapter 2 is the well with the highest NPV of all simulated wells. This chapter will discuss the top wells and their locations in the well on deliverability case and the wellbore file case.

3.1 Optimal well placement for well on deliverability

For this case the Algorithm estimated 118 potential wells with a positive NPV, the associated cell name, NPV and coordinates of the top five is shown in Table 2.

Table 2: Cell name, NPV, XY-coordinates and the depth from surface for the top five best wells in the Hypothetical model using the well on deliverability setup.

Cell name	NPV	X-Coordinates	Y-Coordinates	Depth from surface
249	$13,868,129	5000m	5000m	2250m
289	$13,527,714	5352.8m	5242.9m	2250m
292	$13,302,984	5040.6m	4577.5m	2250m
371	$12,464,989	5000m	5000,	2150m
285	$12,091,012	4946.5m	5378.6m	2250m

As can be seen from the X- and Y-coordinates all of these cells cluster up in the center of the model with a slight variation in depth. The well in cell 249 had the highest NPV of $13,868,129, it is at a significant depth within the system and is in the lowest reservoir rock layer, just above the base rock layer. As the base rock layers have a defined fixed state the NPV of this well may be somewhat

misleading due to the proximity to an endless source of high temperature and pressure, this also applies to 289, 292 and 285 as they are in the same layer and above the fixed state heat source.

The top five best wells are all located in the bottom two layers of the reservoir, right above the base rock, and around the center of the system, right above the heat source. As was mentioned in chapter 2 these wells may have an unreasonably high NPV as their bottom hole pressure is lower than they would realistically have. As all simulated wells had the same well bottom pressure and the flow from the wells is calculated based on the pressure difference between the well bottom and the cell it is to be expected that wells in the lowest layers will have the highest amount of flow, as these cells will have the highest pressure and therefore the largest pressure difference. Simulations using the deliverability setup will, however, give some idea of the location of the optimal wells. These locations (at depth above the heat source) are not surprising and would most probably be some of the first well locations considered as targets for drilling in a geothermal system like this.

3.2 Optimal well placement for well on wellbore file

For this case the Algorithm estimated 24 potential wells with a positive NPV, the associated cell name, NPV and coordinates of the top five is shown in Table 3.

Table 3: Cell name, NPV, XY-coordinates and the depth from surface for the top five best wells in the Hypothetical model using the well on wellbore file setup.

Cell name	NPV	X-coordinates	Y-coordinates	Depth from surface
981	$2,213,888	5000m	5000m	1650m
859	$2,135,811	5000m	5000m	1750m
1225	$1,983,308	5000m	5000m	1450m
1103	$1,940,392	5000m	5000m	1550m
1469	$1,747,260	5000m	5000m	1250m

As can be seen from the X- and Y-coordinates all of these cells line up in the center of the model however in this case they are not at the same depth as in the deliverability case. The well in cell 981 had the highest NPV of $2,213,888.311 and is located in the center of the model, above the heat source, at a depth of 1650m. This location, as well as the other locations in the top five, is within the lower part of the upper heat zone, seen on Figure 3 as the yellow area in the middle of the figure. These locations are to be expected as some of the best locations as they are within a high temperature zone in the middle of the system and would most likely be some of the first locations considered for drilling. Unlike the top five wells in the deliverability case these wells form a line along the middle of the system as opposed to a plane around a certain depth. This is at least in part due to the fact that the bottom hole pressure in the wellbore case will be different for each well thereby not skewing the results in favor of higher pressure/deeper wells.

None of the top wells are located in the bottom layers closest to the heat source in this case. This is because of the conditions in those cells exceeding the wellbore file data set resulting in the simulation being terminated prematurely. Only 243 simulation simulated more than a year in this case and 1237 simulated more than 0 years. The 994 simulations that simulated more than 0 years but less than 1 year all reached equilibrium within the first ten time steps and therefore terminated the simulation printing out the results at time step ten. These cells are all cells with fixed conditions and it is therefore normal that no change takes place there.

4 DISCUSSION

The algorithm has been successfully tested and seems to give reasonable results in the case of the wellbore file although some limitations are apparent. The following chapter will discuss these limitations briefly as well as summarize the project, present conclusions and make suggestions for future work.

4.1 Limitations

While the algorithm does give reasonable results in the case of the wellbore file it does not for the deliverability case. This is largely due to the fact that the well bottom pressure is the same for all simulated wells in the deliverability case. This causes the pressure difference between the cell and the well bottom to grow with depth and causes unreasonably high flow rates at depth and unreasonably low flow rates in shallower wells. This could potentially be fixed by defining the well bottom pressure as a function of depth thereby account for the increased pressure at depth.

The algorithm assumes that well cost grows linearly with depth when in reality it probably grows exponentially, this means that the cost of deep wells will be lower than realistic while the cost of shallower wells may be higher. Setting up a more complex model for the cost of the wells would shift the algorithm more in favor of shallower wells which may be more feasible targets for drilling.

The algorithm treats all wells as potential targets, this means that wells that for one reason or another are known to be unfeasible, for example above ground water level, will be simulated. This will cause longer processing times and may make the algorithm a less feasible choice than it could be.

The assumption that all wells will have the same rejected enthalpy will skew the results towards higher enthalpy wells more than might be realistic. This can also result in a negative revenue stream during a time period if the enthalpy of the geofluid is lower than the rejected enthalpy. Having the reinjection enthalpy defined as a function of the well enthalpy or simply taking a different approach to calculate the energy production, like defining a set amount of generation per unit mass of steam, could fix this problem.

4.2 Summary

The Algorithm utilizes the Python programing language in combination with the PyTOUGH package in order to create and simulate a large ensemble of possible wells in a numerical TOUGH2 model of a geothermal system. It uses these data to evaluate the NPV of all possibilities and return a list arranged in order of NPV. The Algorithm was tested on a simple hypothetical model using the well on deliverability feature as well as a wellbore file from well BJ-14 in the Bjarnarflag geothermal system Iceland. The well on deliverability case successfully ran all simulations and gave the highest NPVs at depth in the center of the reservoir over the heat source, these results are however somewhat unrealistic due to the excessive flow in deep wells. The well on wellbore file case successfully ran 243 simulations and gave the highest NPVs along the center or the model at a depth of 1250-1750m from surface, in the upper heat zone shown on Figure 2.2.

4.3 Conclusion

The Algorithm is functional despite several limitations and can create a list of optimal wells based on the supplied model. This process is (as are all geothermal reservoir models) subject to many assumptions and is highly sensitive to data reliability and availability, this is especially true for the reliability of the model itself. The Algorithm could therefore be a viable tool for decision makers during the well placement process.

4.4 Future work

While the Algorithm works it is far from perfect and a lot of work could go into improving it. An optimization of the Algorithm itself may be useful as well as further utilizing parallel processing for the pre- and post-processing steps. Restrictions on possible well locations to reduce the amount of simulations run would help with cutting down the processing time of the Algorithm, this could include things like a minimum and maximum depth of wells, disallowing wells in cells at fixed conditions or too close to boundaries. Implementing the error associated with each well to give an idea of the risk involved with each location would give valuable data to the decision makers both on what the safest well choices are and indications as to areas that may benefit from further exploration. Adding in more options for the user, for example by having the productivity index be defined for any given area separately similar to how the area a wellbore file applies to is defined. This should allow for more realistic estimation of the optimal wells and their NPV.

Running the Algorithm with a version of TOUGH2 that is coupled with a wellbore simulator would also be a big step forward. That way much of the problems with providing an appropriate wellbore file for the given depth and design of each well could be eliminated. It is important to test the Algorithm with an accepted model of an actual geothermal system in order to further identify potential problems and evaluate feasibility. If the simulation time can be brought down significantly enough, or access to sufficient processing power is available, the Algorithm could potentially be modified to find the optimal combination of multiple wells. This could for example be done by picking the top few wells and running simulations with each of them along with all other possibilities in order to find the best combination. This work could also benefit from adding criteria that identifies wells as close to identical so that simulations that would have two or more wells in close proximity to each other would be deemed unfit for simulation.

REFERENCES

Renewable Energy Policy Network for the 21st Century: Renewables 2016 global status report, REN21, Paris, France (2016)

U.S. Energy Information Administration: Levelized cost and levelized avoided cost of new generation resources in the annual energy outlook 2016, U.S. Department of Energy, Washington, DC (2016)

U.S. Energy Information Administration: Updated capital cost estimates for utility scale electricity generation plants, U.S. Department of Energy, Washington, DC (2013)

Gehringer, M., Loksha, V.: Geothermal handbook: Planning and financing power generation, Energy Sector Management Assistance Program, World Bank, Washington, DC (2012)

International Finance Corporation: Success of geothermal wells: A global study, International Finance Corporation, World Bank, Washington, DC (2013)

Jefferies, B.: Optimal well placement in the Theistareykir geothermal field for the next well in succession, Reykjavík University, Reykjavík, Iceland (2016)

Pruess, K., Oldenburg, C., Moridis, G.: Tough2 user's guide, version 2.0, University of Califronia, Berkley, California (1999)

Lutz, M.: Learning Python, Fifthe Edition, O'Reilly Media, Sebastopol, CA (2013)

Croucher, A.: PyTOUGH user's guide, University of Auckland, Auckland, New Zealand (2015)

Gylfadóttir, S.: Modeling of the námafjall geothermal system: Numerical simulation of response to production and reinjection, Iceland Geo Survey, Reykjavík, Iceland (2013)

Júlíusson, E.: Personal communication, Dec. 15 (2016)

Kipsang, C.: Cost model for geothermal wells, United Nations University: Geothermal Training Program, Reykjavík, Iceland (2013)

Moon, H., Zarrouk, S.: Efficiency of geothermal power plants: A worldwide review, proceedings, New Zealand Geothermal Workshop 2012, University of Auckland, New Zealand (2012)