ECOLE POLYTECHNIQUE
PROMOTION X2010
LOURDAIS PIERRE

RAPPORT DU STAGE DE RECHERCHE

Heat extraction from porous media with supercritical CO_2

NON CONFIDENTIEL

Option:	Énergie du XXIème siècle
Champ de l'option:	Physique: PHY597
Directeur de l'option:	Arnd Specka
Directeur du stage:	Steven Glaser
Dates du stage:	du 8 avril 2013 au 12 juillet 2013

Nom et adresse de l'organisme:

Civil Engineering Department, UC Berkeley
Center for Information Technology Research in the Interest of Society
621A Sutardja Dai Hall (CITRIS Building)
University of California, Berkeley, CA 94720-1758

Abstract

This report presents an experimental and modelisation work about heat extraction in porous media device using supercritical CO_2. This document exhibits how the vessel boundary conditions are taken into account, describes why and how the simulation use a time dependent injection, introduces how is managed the material thermal conductivity gradient and explain how is built the three dimensions rendering. It also illustrates all this work by a presentation of experiments.

This project takes part in enhanced geothermal systems (EGS), a new type of geothermal technologies which does not require natural convective hydrothermal resources.

Those models involve interesting results which should let to have a better representation of a flow in porous media of such an heterogen fluid as supercritical CO_2. However, this whole work cannot be exploited, specially the three dimension rendering results, due to incompatibility between software libraries.

Résumé

Ce rapport présente un travail de modélisation et d'expérimentation sur un dispositif d'extraction de chaleur en milieu poreux par CO_2 supercritique. Ce document explique comment les conditions aux limites du réservoir ont été prises en compte, décrit le besoin de simuler une injection dépendente du temps, montre comment ont été développé des outils afin de prendre en compte des variations de thermoconductivité du milieu, expose comment a été bati un maillage en 3 dimensions et illustre ceci par la présentation d'expériences.

Ce projet s'inscrit dans le cadre des systèmes géothermiques améliorés (EGS) qui ont pour but de constituer un nouveau type de technologies géothermiques, ne nécessitant pas de source de convection naturelle.

Ces modèles produisent des résultats intéressants, qui devraient permettre de caractériser plus précisemment l'écoulement d'un fluide aussi versatile que le CO_2 supercritique dans un milieu poreux. Toutefois, toutes les ressources n'ont pas pu être exploitées, en particulier le rendu en trois dimensions des résultats, pour un problième d'incompatibilité entre les librairies utilisées.

Contents

List of Figures

List of Tables

Introduction

After a formation in Ecole Polytechnique, where I specialized in "21st century energies", I wanted to spend this research internship in a mechanical project linked to energy production. This geothermal project is perfectly part of my expectations. It also meets two of my aims: spend almost four months in an english-spoken country, and to be close to the San Francisco Bay Area dynamism.

Aknowledgement

I want to strongely thanks those person, who empower me to spend a really interesting and fulfilling internship:

- Steven Glaser: Supervisor of my internship, Mr Glaser accepted me to join his team during this 14 weeks. He also paid attention that the internship progress went well.
- Arnd Specka: Option director in Ecole Polytechnique, Mr Specka helped me to find an internship which suited with my expectations.
- Mario Magliocco: Mr Magliocco is working toward his PhD with Steven Glaser in Civil Systems, I thank him for the huge amount of things he taught me with patience.
- Chris Sherman: Mr Sherman is part of Mr Glaser team. He helped me for several computing issues.

I Enhanced Geothermal System

I.1 Overview of the subject

Geothermal energy is a vast resource that, if efficiently utilized, could satisfy the majority of the base load energy demand in the United States. Current commercial geothermal electricity production is dependent on a number of factors including an optimized combination of geological conditions. Enhanced Geothermal System (EGS) is a geothermal power technology which allows heat extraction where the hydrothermal ressources are not optimal. As a matter of fact, the current extraction of geothermal power requires some natural conditions such as a presence of water, a heat flux enable by convection, a high porosity and permeability of the rock. EGS projects involve a transfer liquid supply, which extracts the heat from a well-suited rock.

Mario Magliocco is working towards his PhD with Steven Glaser, exploring the possibilities of using supercritical CO_2 in order to produce an Enhanced Geothermal System. He explores this possibility by both experimentation and numerical simulation in a Civil Engineering laboratory of University of California, Berkeley. I joined this project.

A supercritical fluid is any substance at a temperature and pressure above its critical point. Thus, a fluid heated to above the critical temperature and compressed to above the critical pressure is known as a supercritical fluid.

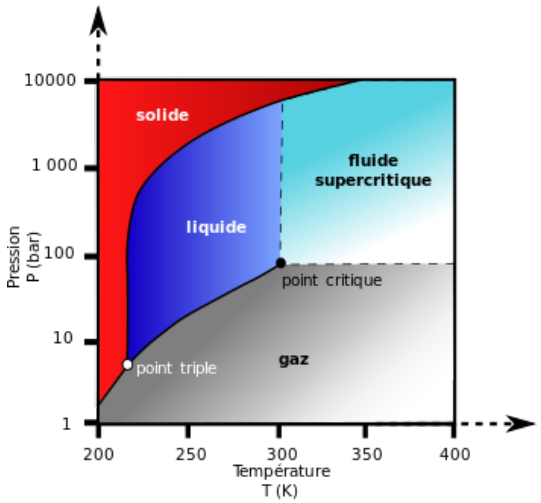


Figure 1: CO_2 different physic states [4].

Besides, as a direct consequence of its definition, a supercritical fluid has properties between those of a liquid and a gas. is property depends a lot on is operating conditions, as we can see on the next figures: the more important variation are around CO_2 critical point ($T_c = 31.1°C; P_c = 73,8\ bar$):

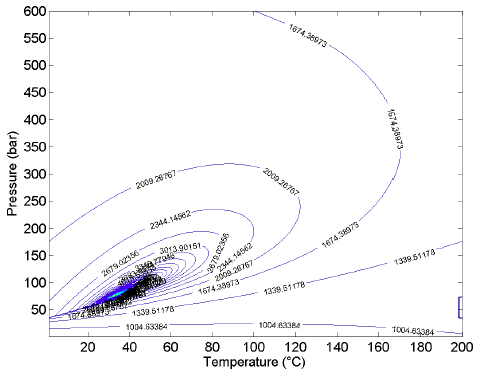


Figure 2: CO_2 thermal conductivity (MATLAB).

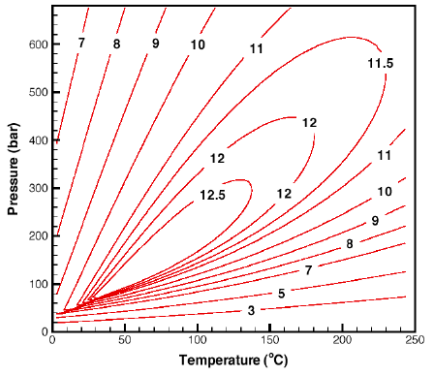


Figure 3: CO_2 mobility (MATLAB).

The new concept of using supercritical CO_2 as the working fluid in EGS for both reservoir creation and heat extraction was first proposed in 2000 by Donald Brown [10]. There are several reasons which tell us to use supercritical CO_2 instead of water or CO_2 [8]. First, the CO_2 thermal proprieties are more fickle than those of water, especially the heat capacity. Then, the mobility of supercritical CO_2 is much higher than $C0_2$ in other operating conditions, as we can see on the previous figures. If we can assume that in a porous media, the Darcy law is appropriate, it implies that the mass flow rate can be higher. Moreover, supercritical CO_2 depend a lot on operating conditions, the difference of density between injection and production points is high and buoyancy forces of supercritical CO_2 which may allow to do not use a pump.

I.2 The experiment

The experiment was already settle by Mario Magliocco. Its purpose is to study the heat extraction in a porous media. The flow occurs in a vessel full of sand, where it is possible thanks to sensors to record temperature in a large number of locations, the mass rate of fluid injection, the injection pressure, the vessel outlet pressure, the accumulation and the pressure difference between the injection and extraction points.

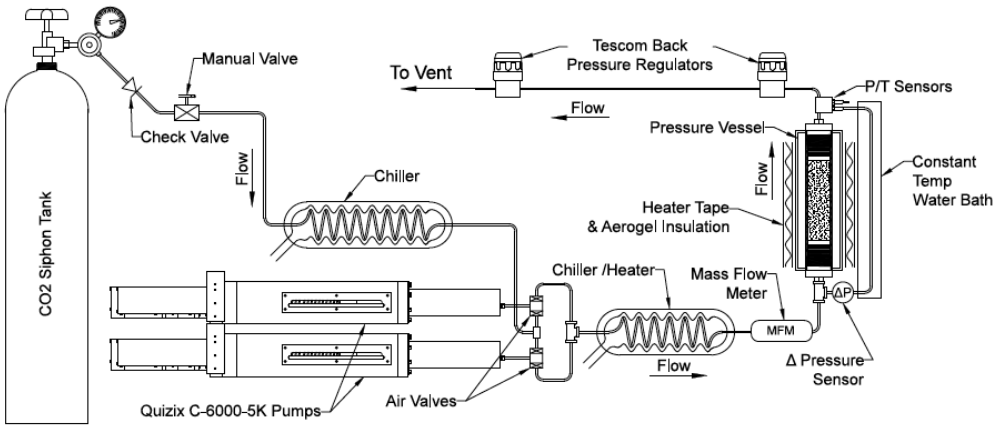


Figure 4: Experiment schema [8].

Vessel length	L = 0.508 m
Vessel diameter	d = 0.451
Quartz density	$\rho = 2650 kg.m^{-3}$
Quartz specific heat	$C_p = 830 J.kg^{-1}.C^{-1}$
Quartz thermal conductivity	K = 8 $W.m^{-1}.C^{-1}$
Permeability	k = 9.3 10^{-13} m^2
Porosity	$\phi = 40\%$
Mean Grain Size	d_{50} = 105 μm

Table 1: Experiment characteristics

The main object of study is a cylinder vessel made of steel, fill with sand.

Figure 5: Experiment devices picture with the insulated vessel (picture).

The grain diameter of the sand used in the vessel is between 53 μm and 105 μm. The following values will be take into account for the analysis[8]

I.3 The numerical simulations

The simulations is realised thanks to TOUGH2, a software released by the Lawrence Berkeley National Laboratory in 1991. Written in standard Fortran77 code, TOUGH2 is a numerical simulator for non-isothermal flows of multicomponent and multiphase fluids[?]. We use it with EOS2 module, which allowed to use both water and CO_2 [7]. As we can the on the following pictures, TOUGH2 use an input file which defines each parameters such as the mesh and the components, and figure out an output file which simulates temperature, pressure and flows in each element of the mesh:

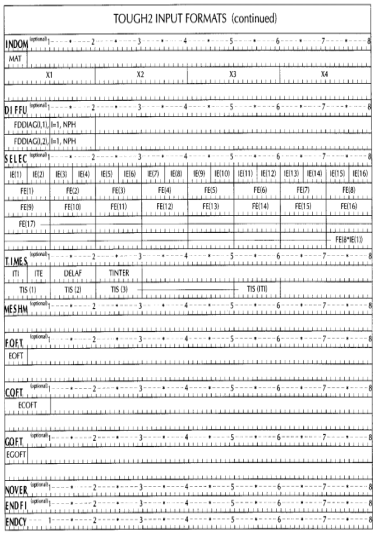


Figure 6: TOUGH2 input file [7].

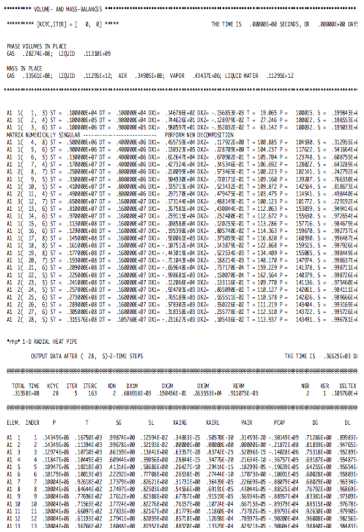


Figure 7: TOUGH2 output file[7].

The input file is hard to configure manually and due to the number of elements, the output file is not easy to analysis. As a consequence, Mario Magliocco wrote a Matlab wrapping to use TOUGH2. The simulation purpose is to compare with the experimental results.

I.4 Internship prospects

When I arrived at the beginning of may, the experiment was not usable. Because of the high mobility of supercritical CO_2, some components as the pumps were leaking. Furthermore, as the used sand is really thin, it was essential to clean some tubes because sand grains escaped from the vessel. As a consequence, it appears that part of my work would be to work with Mario Magliocco to both fixing those issues, and find tricks to prevent the experiments to leak again.

A lot of results have already been collected by Mario Magliocco, and it raised other questions. For example, why is there a vertical temperature gradient at the beginning of the experiment? Does the flow respect Darcy law and, as a result, follow the simulation model? To answer those questions, it seems necessary to simultaneously prospect the literature about porous media, understand how each hardware device works and know how the simulation is built. Working on all those fields was a prospect particularly stimulating.

A Python library also exists to wrapped TOUGH2 simulation [6], an interface including meshing and rendering powerful tools. As we decided to use it for the further experiment, I spent a big part of this internship to modify and apply this library to the experiment.

The main purpose is to get all those tools and modifications ready for new experiment runs before the internship ended.

II The vertical inhomogeneity of Temperature

II.1 Current experimentation

When I started to work on Mario Magliocco's project, I was surprised by the vertical inhomogeneity of temperature at the beginning of the experiment as we can see on the following result graph:

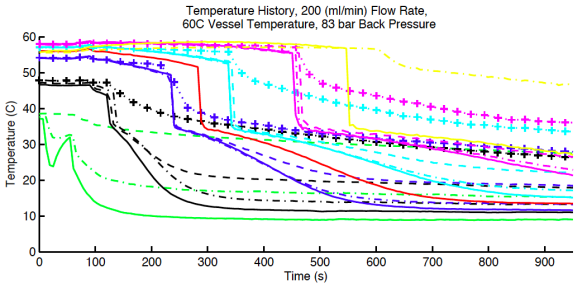


Figure 8: Temperature in the vessel vs time data [8].

In order to explain such a temperature vertical profile, I tried to study how the boundary conditions are settled in this experiments, ans some explanations as convection and gravity effects. Finally, I decided to put surface thermocouples in order to improve the simulation model and try to understand what happens.

II.2 Temperature regulation and boundary condition

There is a thermocouple on the wall of the vessel in order to regulate temperature. Indeed, the vessel is wrapped with glass wool which enable to heat the wall. Naturally, the device is isolated from the exterior with thermal insulation. The vessel is heated each second with a square function, and I wondered if it was enough to get a continuous boundary condition about temperature.

In order to answer this question, I want to get an order of magnitude using Fourrier decomposition and thermal diffusion equation. The wall thickness is h=18 mm, and on the boundary we can assume that the temperture function is a square function with a relative amplitude of α. With a Fourrier decomposition, we want to study the answer to such a function:

$$T(0,t) = T_0 \ * \ (1 \ + \ \alpha \ sin(\ w * t))$$

The adimensionnal temperature is:

$$\Theta(x,t) = \frac{T(x,t) - T_0}{T_0}$$

We get:

$$\Theta(0,t) = sin(wt)$$

According to diffusion law, we get:

$$\Theta(h,t) = e^{-\frac{h}{\delta(w)}} * sin(wt - \frac{h}{\delta(w)}) avec \delta(w) = \sqrt{\frac{2 * D}{w}}$$

The input signal, a square function, is:

$$I(t) = \frac{4}{\Pi} * \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)} * sin((2n+1)wt)$$

The output signal is:

$$O(t) = \frac{4}{\Pi} * \sum_{n=0}^{\infty} \frac{1}{2n+1} e^{-\frac{h}{\delta(w)}} sin((2n+1)wt - \frac{h}{\delta(w)})$$

I draw the graphic thanks to Maple:

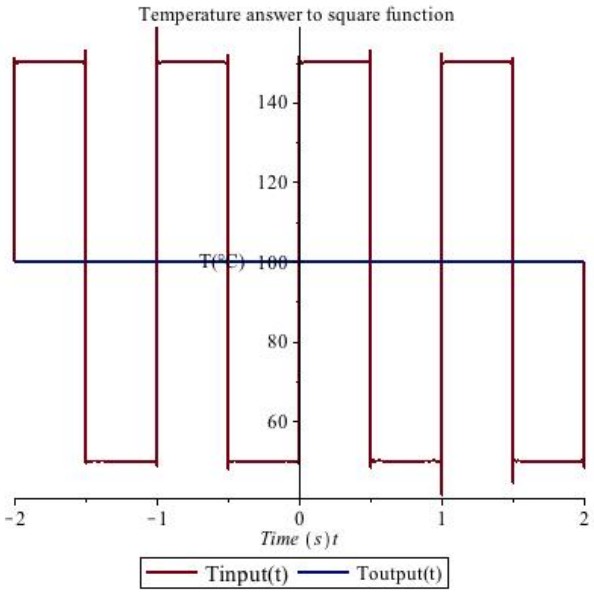


Figure 9: Temperature answer (MAPLE).

Huge assumptions are done here, for example I calculated the diffusion law as if it was a semi infinite wall. However, as the first term relative amplitude is really low ($4.5 * 10^{-13}$ according to Maple calculus), it appears that this assumption is enough to give an order of magnitude. Hence, we can assume that the boundary condition is continuous, equal to 60°C thanks to the regulation system.

II.3 Gravity and convection

. As density depends on operational conditions, we cannot assume that supercritical CO_2 is an incompressible fluid. We want to know if pressure variations due to the compressibility are important, we can wonder if there is a difference in the pressure field in isothermal conditions ($T = T_{up}$), if we process with a spatial discretization. We consider a vertical tube, its length L, only submitted to gravital forces. Two different models:

- In the first case, the pressure field is given by the static fluids law:

$$P(z) = P(L) + (L - z) * \rho_(L) * g$$

- In the other case, we use a finite elements methods, we consider that ρ is a function of both P and T, with boundary conditions:

$$P(L) = P_{up}$$

$$n \ = \ floor(\frac{L}{\delta}) \ , the \ number \ of \ elements$$

and the recurrence equation:

$$P(\delta * (n - (i + 1))) = P(\delta * (n - i)) + \rho(P(\delta * (n - i)), T_{up}) * \delta)$$

which is an approximation of:

$$P(z) = P(L) + \int_z^L \rho(T, P) * g \ dz$$

The Matlab program "*static_pressure_model.m*" return the comparison between those two models.

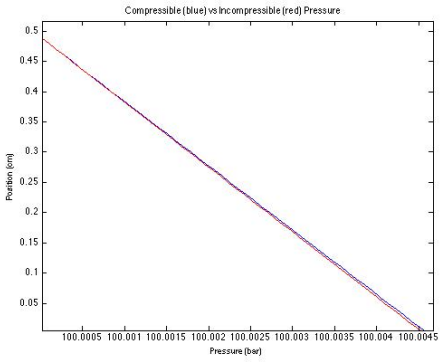


Figure 10: The two models. Operating conditions:P_{up}=100 bar and T_{up}=60C (MATLAB).

In order to be more accurate, we defines a Matlab function which returns a contour plot. The difference is represented as a function of P_{up} and T_{up}.

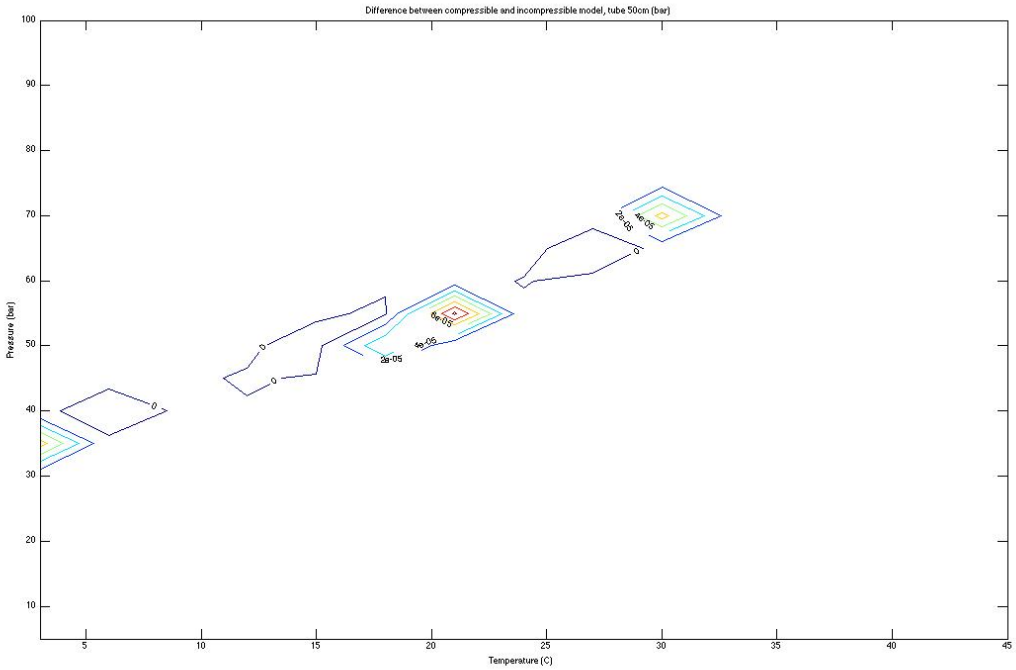


Figure 11: Contour plot of difference between two models (MATLAB).

We can recognize the liquid-gaz coexistence boundary. The uppest point is the critical point. If we stay away from this border - pressure above 100 bar for example - we can neglect the compressibility in isothermal conditions.
Differences between compressible and incompressible models exist around the transition liquid-gaz, but are very small. As a consequence, it seems that the incompressible model is precise enough to get the pressure field in a 50cm tube. However, maybe that assumption is not true when operational conditions are not isothermal.

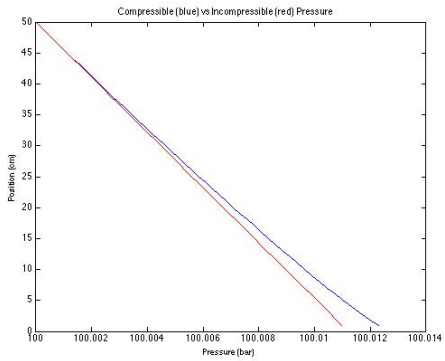


Figure 12: Non-isothermal simulation (MATLAB).

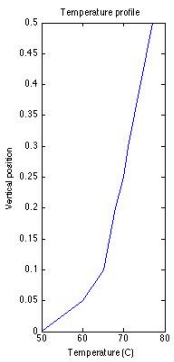


Figure 13: Temperature profile (MATLAB).

In this case, differences appear. It is not important enough to explain the temperature gradient in the vessel, but it show that gravity effects must be taken into account even for a 50cm tube.

What about convection in the vessel? If there is convection,temperature tends to be homogenous but is it possible? Rayleigh number must be calculate in the specific case of porous media [9] [3]. In fact, the Rayleigh number in a porous media is:

$$R = \frac{\alpha * \beta * g * K * d^2}{\kappa * \nu}$$

β is the imposed temperature gradient, it is negative if T(L)>T(0). The radial gradient is really low. As a consequence, at the beginning of the experiment, convection is not possible. It seems only to be a diffusion problem.

In order to understand better what happen, I decided to monitor the temperature along the wall with surface thermocouples. We would be able to check the vertical profile of temperature all along the vessel, and their variation with time.

In order to be sure that there is not convection in the model, I will try to implement a 3D mesh in the finite element method. Actually the mesh used is a 2D mesh with cells wich respect the volume of axis-symmetry. However, we can imagine convection cells which broke the axis-symmetry, it is essential to build a 3D mesh in order to know with simulation model if such a convection exist.

II.4 Improvement of temperature monitoring

We want to take those boundary phenomenon into account during the experimentation. As a consequence, I decided to use surface thermocouples on the wall of the vessel. Those thermocouples are supposed to enable us to determine the heat exchanges, and to set boundary conditions on the numerical simulation.

Those thermocouples, distibuted by OMEGA company, are T-type, which means that they are made of Copper Constantan. They can be used from -60°C to 175°C.

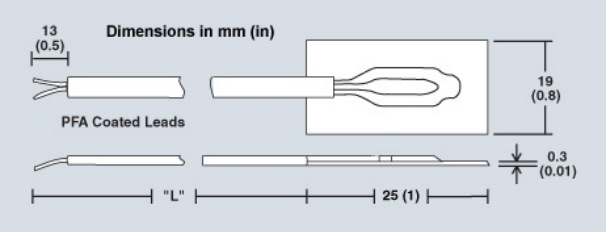


Figure 14: Surface thermocouple schema [2].

Thanks to T-type connectors and T-type wires, the datas can be import to the computer. Using Labview software (Microsoft), those values are stored as text files and can be used during the post-processing.

Thermocouples	Target location	Real location	MConn.	FConn.	Channel	Input
A	0	0	A	A	5	25
B	5	5.3	B	B	7	27
C	10	9.7	C	C	8	28
D	15	14.7	D	D	9	29
E	20	19.8	E	E	10	30
F	25	24.6	F	•	•	•
G	30	29.6	G	G	11	31
H	40	39.5	H	H	12	32
I	50.8	50.5	I	I	13	33
J	top cap	58.7	J	J	6	26

Table 2: Surface thermocouples characteristics

We ordered 10 thermocouples, I had to determine the position. As we did not notice any experimental issue concerning the axis-symmetry, I decided to put them on the same angle from the vessel axis. Naturally, we want to collect datas at the same vertical position than those inside the vessel. I also wanted to get the temperature at the top and at the bottom of the vessel, on both caps. To define the other location, I used Mario Magliocco's previous results: If we look for the initial conditions, the temperature gradient versus

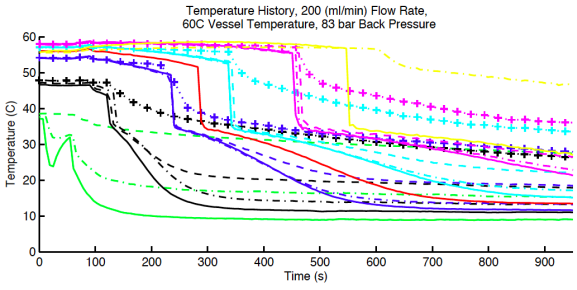


Figure 15: Temperature in the vessel vs time data [8].

vertical location appears more important at the bottom of the vessel. As a consequence, we will try to put surface thermocouples all along the vessel but more focused at the bottom of the vessel. Here are the target and real positions for the thermocouples, these information are essential in order to import the datas:

Location are give in centimeters. Connector letters and channels are not suited because I cut some wires during the settings.

This is the experimental device:

Figure 16: Experimental Disposition (picture).

The wires came through a PVC pipe because they are really thin and weak.

The initial datas are imported in the python wrapping of TOUGH2, in order to set temperature in 8 wall mesh locations. Thanks to a linear interpolation, conditions are given to all the wall boundary. Moreover, the top and bottom surface thermocouples (A and J) give boundary conditions of both caps. We do not use the value during experiment to change the numerical simulation datas, as we do for inside thermocouples. However we can compare those datas in order to understand heat exchange.

III Time dependent simulation

III.1 Gibbs effect

The simultion model given by TOUGH2 and Mario Magliocco's result seems to be highly correlated, thanks to the graphic he figured out:

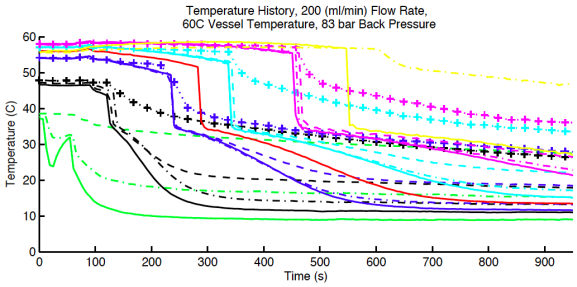


Figure 17: Experimental datas (hashed) vs Simulation Model (Plain) [8].

However, the curves at the beginning of the experiment does not seem totally well-fitted, especially the simulation green curve reach a temperature below the minimum temperature of the experiment, which does not appear acceptable. Furthermore, the black curve is supposed to show an inflexion point, whereas the curvature does not fit. I though it was an initial condition issue. Indeed, the green curve correspond to the inlet thermocouple. In the simulation model, the temperature initial condition is given by the experiment initial condition which is around 55°C. However, the temperature just after is supposed to be the injection temperature which is around 15°C. As a consequence, the simulation model has to deal with a jump discontinuity of temperature, and I though it generates a Gibbs phenomenon, because the model try to do a modelisation with a finite sum of continuous function.

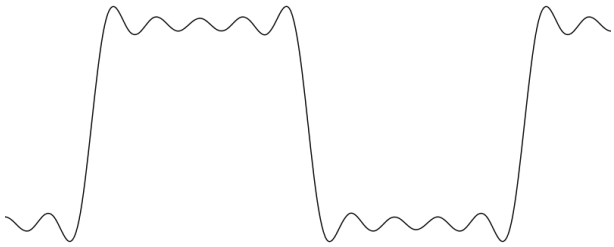


Figure 18: An example of Gibbs phenomenon: modelisation of square function with a finite number of Fourrier function [1].

Concerning the black curve, I was not able to give an explanation or predict a modification, so I stayed focused on the green one. But as it is a thermocouple near the injection point, so I hoped it was linked with the same issue.

TOUGH2 enable to use a time-dependent simulation. I tried to use this function, it requires to specified the specific enthalpy of the fluid for mass injection. Using CO_2 thermodynamic tables, I implemented the Python wrapping of the Fortran TOUGH2 code in order to figure out the difference between those two models. However, I need to define what values of injection rates vs time can fit the experiment. As a consequence, I studied the flow in this porous media, which enabled me by the same occasion to wonder if the Darcy law is valid in this vessel.

III.2 Darcy law

TOUGH2 solves Darcy law equations in order to give a numerical model. The Darcy law gives a relation between the mass flow rate and the drop pressure over a given distance. According to Dynamics of Fluids in porous media, it can be used if the Reynolds number is lower than 10 [3]. We want to determine the maximum flow rate which enable to use the Darcy law. It depends a lot on the operating conditions, as the fluid is supercritical CO_2.

$$Re = \frac{\rho * q * d}{\nu}$$

Where:

- ρ is the fluid density.
- q is the local specific discharge.
- d is a length dimension, I use d_{50}, a characteristic grain diameters already measured for the used sand.
- ν the dynamic viscosity of the fluid.

In order to calculate the Reynolds number, I made the two following assumptions:

- The vessel is filled with an homogeneous porous media.
- Radial variation are not too important (valid assumption thanks to Mario Magliocco's former results), which means that ρ and the velocity only depends on the vertical position.

The accumulation has to be taken into account as it change the mass flow rate conservation. With an homogeneous accumulation, I get the following Reynolds:

$$Re\ (P, T, Q_{pump}, Q_{out}) \ = \ \frac{d_{50} * \rho(Q_{pump} + \frac{(Q_{pump} - Q_{out}) * z}{L})}{\phi * A * \mu(P, T)}$$

Where:

- Q_{pump} is the volumetric flow rate output of the pump, Q_{out} is the volumetric flow rate output of the vessel.
- z is the vertical position, L is the length of the vessel
- ϕ is the porosity, for the vessel: $\phi \ = \ 41\%$
- A is the vessel cross section

However, we can use only one Coriolis mass-flow meter, it is not possible to get both Q_{pump} and Q_{out}. The maximum flow rate is higher with accumulation, so I calculate the minimum maximum flow rate which enable to use the darcy law, without accumulation ($Q_{pump} = Q_{out}$)

$$Re\ (P, T, Q_{pump}) \ = \ \frac{Q_{pump} * \rho * d_{50}}{\phi * A * \mu(P, T)}$$

I used matlab in order to display a contour graph of massic flow rate Q_{max}, defined by:

$$Q_{max} = \{Q \backslash Re(T, P, Q) = 10\}$$

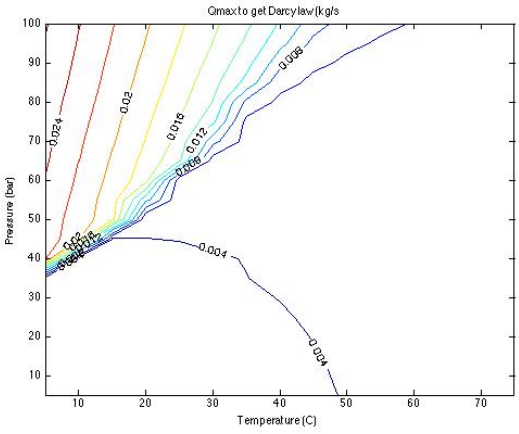


Figure 19: Q_{max} to use Darcy law (MATLAB).

In our operating conditions, Q_{max} is included between $4g.s^{-1}$ and $6g.s^{-1}$, whereas our massic injection flow rate is around $2g.s^{-1}$ (exactly $1.6025g.s^{-1}$ for the results on the previous graphic). As a consequence, Darcy law can be used during this experiment and TOUGH2 simulation should be valid.

Now we have to determine what are the value of time-dependent flow rate injection. We want to prevent from a discontinuity drop, thus the initial mass flow rate must be equal 0. We can use the fluid velocity, which I estimated around 1 $mm.s^{-1}$, to make an estimation of the time dependent flow rate. We can also use the pump experimental datas to have an idea how to change the model.

III.3 Results

I have tried several time-dependent injection. The following table gives he evolution of the flow rate for three of them. Those evolutions are linear.

Here are the figures, plotting the experimental curves versus the simulation result:

Time (s)	Simulation 1	Simulation 2	Simulation 3
0	1.6025	0	0
50	1.6025	1.6025	0.4
100	1.6025	1.6025	0.801
200	1.6025	1.6025	1.6025
5000	1.6025	1.6025	1.6025

Table 3: Simualation characteristics

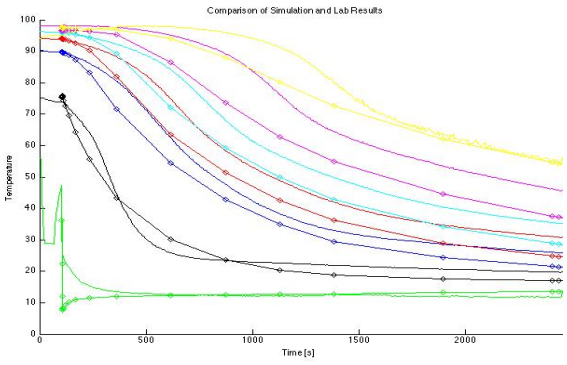


Figure 20: Simulation 1 (MATLAB).

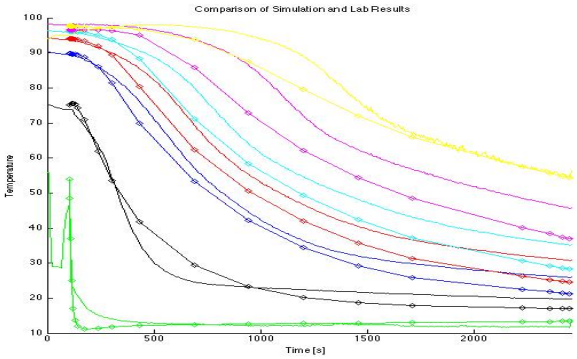


Figure 21: Simulation 2 (MATLAB).

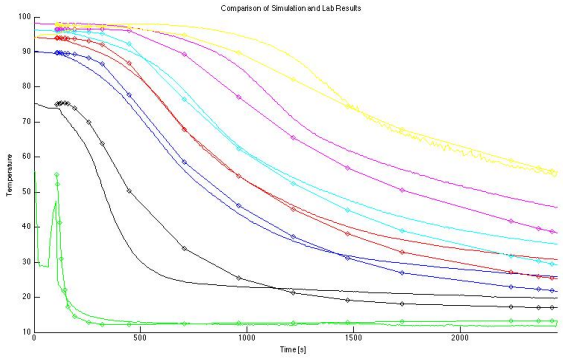


Figure 22: Simulation 3 (MATLAB).

Let's study especially the green curve (input thermocouple). On the previous simulation (Figure 18), it is not differentiable with a singularity point just after the beginning of the experiment. Obviously, it is not realistic, and the temperature is below the lowest temperature of the experiment (injection temperature). On the second graphic, the temperature is still below the lowest temperature, but the singularity point does not exist anymore. On the third graphic, the two curves are really closed, there is not any singulary point, and the temperature stay superior or equal to the injection temperature.

The black curve corresponds to another thermocouple, at the centre of the vessel, 10 centimeters above the input thermocouple. On the first figure, the simulation curve is concave whereas the experiment curve get an inflexion point. The second and third graphics show curves with inflexion, but the distance between experimental and numerical curves are higher.

Concerning all the other thermocouples, curves are always similar (with an inflexion point), but experimental an numerical curves seem closer on the third figure.

To conclude, as the third graphics is definitely the closest to the experimental data, taking into account a time-dependent injection concernig the simulation is efficient. However, does it make any sense to consider the flow reach the maximum value after 100 seconds? Although the experiment results show that this time lapse is around 40 seconds, I do not want to use this value. The simulation must use only initial data, if we use other experiment results it does not make a lot of sense to compare experiments and simulation. I would prefer to build a transition time of injection with relevant characteristic numbers.

I get an estimation of fluid velocity with target flow value:

$$v = \frac{Q_m}{\rho(T_{injection}, P_{injection}) * A * \phi} \approx 1mm.s^{-1}$$

The vertical range of perturbation area is about the difference of elevation between level 0 thermocouples and level 1 thermocouples: D=10 cm. Thus, the fluid need about the following time to reach a permanent state:

$$t_{delay} = \frac{D}{v} \approx 100\ s$$

I am aware that this reasonment is not totally scientific: I built a characteristic number after using it in a simulation. However, I just wanted to understand if a 100 seconds injection delay for this 5000 seconds experiments sounds crazy, and it does not seem to long thanks to t_{delay}.

IV Pytough implementation

IV.1 Why use it?

PyTOUGH (Python TOUGH) is a set of Python software routines for making it easier to use the TOUGH2 geothermal reservoir simulator. Using PyTOUGH, it is possible to automate the creation and editing of TOUGH2 model grids and data files, and the analysis and display of model simulation results.
Chercher sur les premières page de Pytough Easier to change configuration Powerful tools: - 3D Mesh - Thermal conductivity wrapping
All the code in annexe

IV.2 3D Mesh

Here is a cutaway view of the vessel I tried to import in TOUGH2:

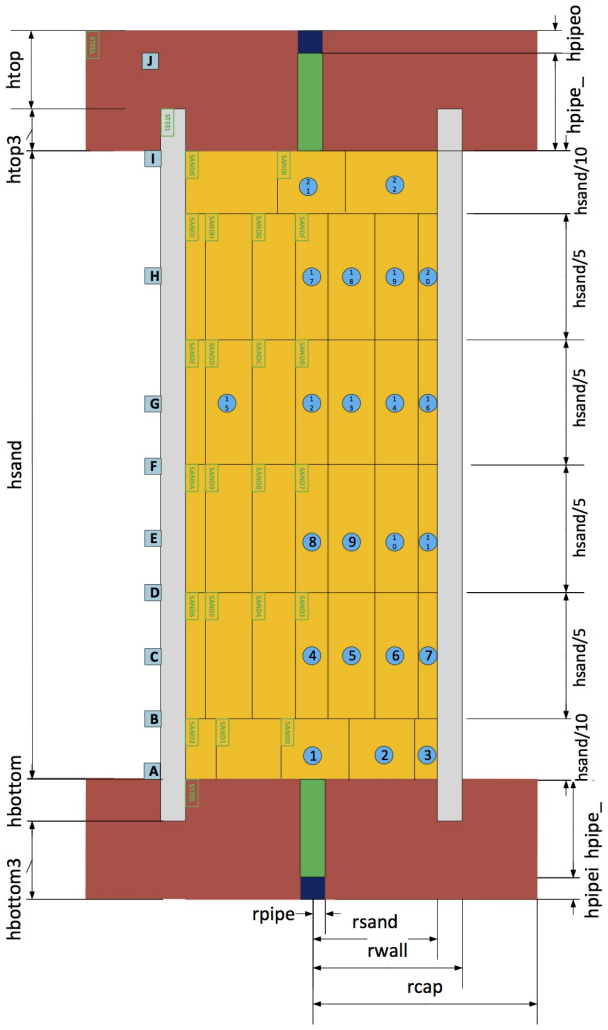


Figure 23: Modelisation target (Microsoft VISIO)

It includes:

- 3 cylinders (top, middle, bottom): thanks to functions I implement in Python, it is possible to create nodes, columns, layers and connections to get a 3D cylinder mesh. It is also possible to connect cylinders together, and to create hollowed cylinders. It is possible to create most of the 3D axis symmetric mesh. The vessel is made with 3 cylinders, connections between the middle one and the others have been built.
- 5 different media (steel, sand, pipe input, pipe output, pipe)
- 21 different area of sand: it enables to take into account the temperature differences, and the thermal conductivity differences. This spatial discretization enables to use datas around each thermocouples.
- 22 inside thermocouples
- 1 injection point: It is a time-dependent injection point as we have seen in the previous part.
- 1 sink at the top of the vessel.

This is a 3D Mesh. Although the previous simulation - 2D which take into account that elements volume depend on their distance from the axis - is really faster to compute, this one must be more accurate concerning the convection phenomenon for example.

All the parameters can really be easily changed. If we add some thermocouples, we just must add it to a dictionary. If the geometry changes but remains axi-symmetric, the mesh can easily be defined. However, a five points injection system would be a next experiment, but it would be a lot more difficult to do the mesh with the functions I implemented.

IV.3 Interpolation

We need to use an interpolation in order to get the initial conditions in each element of the vessel. Actually, TOUGH2 is able to determine temperature in each element of the mesh, but we cannot define each temperature with only the thermocouples. We use thermocouples Mario Magliocco already put in the vessel in order to get the profile temperature of some points:

Vertical \ Radial x	0	$\frac{r}{3}$	$\frac{r}{2}$	$\frac{2*r}{3}$	r
0	X	•	X	•	X
$\frac{h}{5}$	X	X	•	X	X
$\frac{2*h}{5}$	X	X	•	X	X
$\frac{3*h}{5}$	X	X	•	X	X
$\frac{4*h}{5}$	X	X	•	X	X
h	X	•	X	•	•

Table 4: Thermocouples positions

want to obtain each temperature of the 30 positions, whereas there are only 21 sensors. It seems relevant to get them by an interpolation, we use sensors from the same vertical position. It is an axi-symmetric experiments. We do an assumption: there is only diffusion exchanges. Although we think there is convection, the main heat exchange phenomenon is diffusion, as we have already calculate the Rayleigh number. For a vertical position, we use the following equation:

$$T(r,z) \;=\; A(z)\; ln(r) + B(z)$$

Then, we use a Python function, *numpy.intepr2d* - linear interpolation a a 2D-mesh- in order to get all the initial conditions.

IV.4 Thermal conductivity Wrapping

After his former experiments, Mario Magliocco think that there is a difference between the TOUGH2 simulation and the experiment, mainly due to the difference of thermal conductivity of the sands. As the temperature is not uniform in the vessel, the thermal conductivity of the sand is not exactly the same. In most of the uses, it is not really important because those variations are not really important. However, as we use supercritical CO_2, the fluid depends a lot on the operating conditions. To verify those assumptions, we should take those variations into account.

TOUGH2 does not allow us to change the thermal conductivity of the rock type. As a consequence, I tried to implement a Python wrapping which takes for granted the thermal conductivity variation of the rock type, thanks to both a time discretization and a space discretization.

- Time discretization: The whole period of the experiment (around 5000 seconds) is divided in several time steps. Each time, the initial conditions are given by the previous step. TOUGH2 run a simulation for the time of the step, and then data are exported.
- Space discretization: Around each thermocouples (21), I defined a different rock type even if all of them are sand. The only difference is the thermal conductivity. This information is defined at the beginning of each time step, thanks to the data collected at the end of the previous step. In order to collect the information of the first step, I used the initial data of the thermocouples.

A fonction enable then to gather all the collected data in one file. Then a Python class I created, *Experiments.py* , enable to plot both the simulation results and the experiments one.

IV.5 Availibility of the code and documentation

Some errors still occurs in this code. The VTK package in order to show a rendering cannot function in current Operating System. I have tried several methods in order to install this package, but I did not succeed.

In order to keep this code available, I put everything on my GitHub serveur, on the following link:
https://github.com/pierrelourdais/ModifiedPytough.
I write a documentation on this serveur, *README.md*. I hope this code can be usefull for the following experiments in this laboratory. All the code is also explained in ANNEXE A and ANNEXE B of this document.

V Experiments

V.1 Settle the experiment

I mostly did two things during this internship to improve the installation: set in place the surface thermocouples, and help Mario Magliocco to fix some parts of the experiment. CO_2 pumps were leaking, and it was essential to come apart them to clean it and replace some rings. Some sands left the vessel, and it involves some working errors. Besides pumps were not efficient enough, the back-pressure regulator was not able to work because of the sand. After come apart those components, we decided to add filters on several point of the experiment.

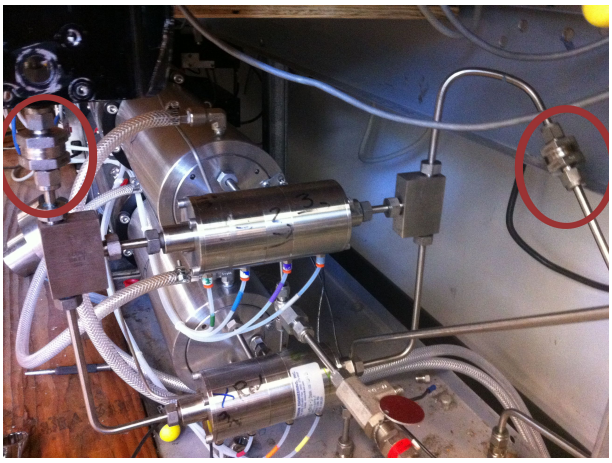


Figure 24: Two filters were added near the pumps valves (picture)

As Mario already used the Coriolis mass flow meter in order to verify the pump delivery, I change this component position and put it at the top of the vessel. Instead of getting the injection datas, it will help us to measure CO_2 accumulation in the porous media.

V.2 Problems during the simulation

VTK is a rendering library which was released in 1998, and often uploaded. This library, which enables 3D computer graphics, modeling, image processing and volume rendering, is essential to run a PyTOUGH simulation rendering.

I have tried for nearly two weeks to download different versions, to launch it on several operating systems and to build the library with different options. It seems impossible to launch this library on a current operating software, and it does not make a lot of sense to run it on a previous one because of the compatibility of all the other devices. If I would stay longer on this project, I would have tried to change PyTOUGH rendering classes in order to use more common rendering package.

V.3 Problems during the experiments

Even if the pumps have been washed, some errors still remains. The two pumps behavior is not the same, especially there is a pressure decrease in one of them. The pipes have been screwed again, . We also find sand in the back pressure regulator, whereas it was supposed to be clean and we put filters. Because of all those issues, we were not able to run an experiment before the end of the internship. During the last week, we will try to remove all the sand, put new filters at the bottom of the vessel and settle again the thermocouples in the right locations.

However, we tried to calibrate the thermocouples and to heat the vessel, it seems to work. Those sensors are ready for further experiments.

VI Ecole Polytechnique formation

VI.1 A full apllied project

To perform in such a laboratory, it is essential to gathered three different kind of backgrounds:

- Be able to understand theorical models
- Be able to settle an experiment
- Be able to programm

Ecole Polytechnique formation give me several clues to face it, however such an internship requires versatility, and I was missing some skills in those three domains. However, it is really fulfilling to learn aptitudes simultaneously in theoretical models, experiment implementation and computing sciences.

VI.2 Understand theoretical models

Even if I followed solid and fluid mechanical lessons in Polytechnique, I had to deal with a new kind of material: porous media. I read some documentation during my first weeks, and often have to come back on this bibliography all along this internship.

VI.3 Settle an experiment

There were different types of manipulations. At first, I must took apart different devices in order to clean and improves some devices. Each changement, even small one, took me quite a long time as it required to change a bigger part around it. As example, when I moved the Coriolis mass flow meter position, it also requires to create several pipes and new supports. Concerning the surface thermocouples, it required to anticipate in order to settle it. However I had to deal with some mistakes, for example I cut 3 thermocouples wires, it took a long time to fix it.

I think it was the domain I was the weakest, as I did follow only few experimental lessons in Ecole Polytechnique.

VI.4 Programming

During this internship, whereas I used to program only on Matlab, I tried to develop my computing science skill, I had to deal with the following languages:

- Matlab: running and post-processing simulations
- Python: another way to run and post-processed simulations, more object-oriented.
- Fortran: native language to the TOUGH2 software.
- Unix command: a way to administer Berkeley server and install all the required libraries.

Last year I at Polytechnique concerning the implementation of the Finite Element Method (MEC557) at Polytechnique, this lesson was really useful during my internship. I tried to reproduce the same kind of reasonment when I wrote the Matlab and Python modules.

Conclusion

Geothermal energy represents a considerable potential of sustainable and environmentally friendly energy available worldwide. But exploitation of this source of energy thanks to traditional systems requires the combination of several specific field conditions which meet only few times in nature. That is why the proposition of a novel renewable energy concept based on heat supercritical CO_2 for both reservoir creation and heat extraction seems to be an opening for the geothermal energy sector. These new geothermal technologies, called Enhanced Geothermal Systems (EGS) should permit to generate energy and electricity on fields which were not usable. As high-performance EGS would permit a dramatic development of geothermal energy in future decades, this outlook seems really encouraging.

However, assessment of the potential of EGS operated with supercritical CO_2 is at an early stage. Even if simulations have shown highly promising results, the research to date on this technology is still limited. Laboratory experiments and field tests have to be set up in order to validate the first results coming from numerical simulations.

The present report is focused on some points which are supposed to improve the experiments and the simulation. It shows how we took into account the boundary conditions, how important it is to set smooth inputs with TOUGH2 software, and how deal with temperature inhomogeneities.

However, part of the work is not fully usable, as I was not able to run the VTK rendering library. Further work could be to change the PyTOUGH native code in order to use other libraries, or create a rendering library.

References

[1] Gibbs phenomenon example. http://en.m.wikipedia.org/wiki/File:Gibbs_phenomenon_10.svg. Accessed: 2013-05-03.

[2] Surface thermocouple schema. http://www.omega.com/toc_asp/sectionSC.asp?section=A&book=temperature. Accessed: 2013-05-28.

[3] Jacob Bear : *Dynamics of Fluids in Porous Media*. Dover, 1988.

[4] Mark Jacobs Ben Finney : P/t co2 diagram, novembre 2010.

[5] Adrian Croucher, 2011 : PyTOUGH: a Python scripting library for automating TOUGH2 simulation.

[6] Adrian Croucher, 2012 : PyTOUGH user's guide, Version 1.3.5.

[7] Curt Oldenburg Karten Pruess et George Moridis : *TOUGH2 User's Guide Version 2.0*. Berkeley Lab, 1999.

[8] Mario Magliocco et Steven Glaser, 2013 : Laboratory and Numerical Studies of Heat Extraction from Hot Porous Media by Means of supercritical CO2.

[9] D.A. Nield, 1968 : Onset of Thermohaline Convection in a Porous Medium.

[10] Karsten Pruess, 2006 : Enhanced Geothermal System Using CO2 as Working Fluid.

Annexe A: Modifications in PyTOUGH

The PyTOUGH library enable the user to control potentially all aspects of a TOUGH2 (PRUESS, 2004) simulation. It was released by the University of Auckland. I modified and creates some class of the library in order to support our experiment. All this code is available on GitHub: https://github.com/pierrelourdais/ModifiedPytough.

Mulgrid.py

This class enable to create a geometric structure. Here are the functions I created:

- *create_cylinder*(): enable to create a mulgrid object of cylinder. Return an objects with nodes, columns, layers, and connection on the main axis, with the following parameters:
  - self: (mulgrid) A mulgrid object.
  - hmin: (float) Vertical position of the lower node.
  - length: (float) Length of the cylinder.
  - radii: (float) Radius of the cylinder.
  - theta_step: (int) Number of step. Maximum number: 62.
  - layer_step: (int) Number of vertical step. Maximum number: 62.
  - radii_step: (int) Number of radial step. Maximum number: 62.
  - nameofcylinder: (str) One letter. Each elements is name with 3 characters: 'numberinthelayer'+'nameofcylinder'+'nameoflayer'.
  - filename: (str) The built cylinder is written in a 'filename.dat' file.
- *substract_cylinder*(): enable to substract a cylinder to a mulgrid cylinder. The new cylinder is scooped out, with the folowing parameters:
  - self: (mulgrid) A mulgrid object.
  - radii: (float) Radius of the hole. Must be inferior to the main cylinder, or return an empty mulgrid.
  - theta_step: (int) Number of radial step of the main cylinder.

T2grid.py

This class enable to create a grid structure. Here are the functions I created:

- *create_cylinder_connection*: enable to create radial connections between 2 t2grid cylinder objects. This function must be used when a cylinder is inside a scooped out one. The following argument must be used:
  - self: (t2grid) The main t2grid object which includes the 2 cylinders t2grid objects.
  - smaller_grid: (t2grid) The t2grid object of the smaller cylinder.

    * larger_grid: (t2grid) The t2grid object of the larger cylinder.
    * radii_step_int: (int) Number of radial steps of the smaller cylinder.
    * layer_step_int: (int) Number of vertical steps of the smaller cylinder.
    * radii_step_ext: (int) Number of radial steps of the larger cylinder.
    * layer_step_ext: (int) Number of vertical steps of the larger cylinder.
    * theta_step: (int) The common theta_step of both cylinders mulgrid objects.

- *create_cylinder_connection*: enable to create vertical connections between 2 t2grid cylinder objects. This function must be used when a cylinder is just above another one. The following argument must be used:
    * self: (t2grid) The main t2grid object which includes the 2 cylinders t2grid objects.
    * upper_grid: (t2grid) The t2grid object of the upper cylinder.
    * lower_grid: (t2grid) The t2grid object of the lower cylinder.
    * radii_step_up: (int) Number of radial steps of the upper cylinder.
    * radii_step_low: (int) Number of radial steps of the lower cylinder.
    * rayon: (float) Radius of the smaller cylinder.
    * theta_step: (int) The common theta_step of both cylinders mulgrid objects.

T2data.py

This class enable to write all the TOUGH2 inputs in a .core file. Here are the functions I modified:

- *__init__*: define a t2data objects. Incon have been modified in order to get to initial datas per elements: pressure (incon1) , and temperature (incon2, created).
- *write_incons*: enable to write the 2 conditions in the .core file.

Step.py

I create this class. It is a wrapping to run TOUGH2 during a step, in order to proceed a time discretization. Here are the functions I created:

- *__init__*: define a step objects, with the following parameters:
    * self: (t2grid) The main t2grid object which includes the 2 cylinders t2grid objects.
    * name: (str)
    * previousstepcsv: (file)
    * infileexperiment: (file)
    * time: (np.array)

    * outfile: (file)
    * outfilecsv (file)
    * vessel (t2data)
    * resultat (dict)
    * thermocouples (dict)
    * thermocouplesw (dict)
    * mulgrid (mulgrid)

- *setinitialT*(): Uses experimental file to import thermocouples datas and fill thermocouples dictionary.
- *setinitialTw*(): Uses experimental file to import surface thermocouples datas and fill thermocouplesw dictionary.
- *importpreviousnumericaldatas*(): Imports a .csv file, build two dictionaries: thermocouples and thermocouplesw.
- $mesh\_definition()$: Defines the well-fitted mesh.
- $vessel\_definition()$: Defines the t2data object with the well-fitted parameters.
- $rocktype\_definition()$: Defines the rock types, especially the different kind of sands with the correct thermal conductivity values.
- $parameters\_definition()$: Defines the well-fitted parameters.
- $generation\_definition()$: Defines the generation rate. It can depend on time as we have seen previously.
- $incon\_definition()$: Defines the initial condition values, thanks to the previous datas (step 1 and more) or the experiment file (step 0)
- $write\_vtk(vtkfile)()$: Draw the mulgrid object thanks to VTK library.
- *write*(): Write the input file (".core) to run Fortran executable TOUGH2.
- *run*(): Run Fortran executable TOUGH2
- *exportnumericaldatas*(): Import the Fortran output file (.listing), import as a dictionary and the export as a .csv file.

Experiment.py

This class concern the experiment post-processing. Here are the functions I created:

- *importnumericaldatas*: Imports datas from the experiment files.
- *plot*: Plots experiment datas versus simulation ones.

Other

Those function are not built in a class, but are used in the main module (main.py) Here are the functions I created:

- *initial_interp*: Use a logarithmic interpolation according to IV.3, then use numpy.intepr2.
    * position: (list) The position you want.
    * thermocouples: (dict) Thermocouples datas, including positions and temperature values.
- *initial_interpw*: Use a linear interpolation along the vertical axis.
    * position: (float) The vertical position you want.
    * thermocouplesw: (dict) Surface thermocouples datas, including vertical positions and temperature values.
- $findindiceblock$: Given a position in the vessel, it returns the closest element.
    * position: (list) The position you want.
    * gridblock: (t2block) The grid containing the elements.

Annexe B: Modelisation of the vessel experiment

Here is the main code:

```
import math
import numpy
from mulgrids import *
from t2data import *
from t2grids import *
from t2incons import *
from t2listing import *
from t2thermo import *
from copy import copy
from matplotlib.mlab import griddata
import matplotlib.pyplot as plt
import vtk
import scipy.interpolate

Tinjection=20
nombre_step=2
timetotal=2000
timestep=float(timetotal)/nombre_step

for compteur in range(nombre_step):
    name='step_'+str(compteur)
    outfile='step_'+str(compteur)+'.listing'
    outfilecsv='step_'+str(compteur)+'.csv'

    if compteur==0:
        previousstepcsv=None
        time_inj=[]
        rate_inj=[]
        for i in range(10):
            rate=1.6e-3*float(i)/10
            time_inj.append(float(timestep)/10)
            rate_inj.append(rate)
        infileexperiment='6.18.2013_11.07 AM.txt'

    else:
        previousstepcsv='step_'+str(compteur-1)+'.csv'
        infileexperiment=None
        time_inj=[]
        rate_inj=[]
        for i in range(10):
            time_inj.append(float(timestep)/10)
            rate_inj.append(1.6e-3)
    current_step=step(name,previousstepcsv,infileexperiment,timestep,outfile
    ,outfilecsv)
    current_step.vessel_definition()
    current_step.mesh_definition()
    if compteur==0:
        current_step.setinitialT()
        current_step.setinitialTw()
    else: current_step.importpreviousnumericaldatas()
```

```
        current_step.rocktype_definition()
51      current_step.parameters_definition()
        current_step.generation_definition(rate_inj,time_inj)
53      current_step.incon_definition()
        current_step.write_vtk('vtk') #dont work
55      current_step.write()
        current_step.run()
57      current_step.exportnumericaldatas()
    current_experiment=experiment()
59  current_experiment.gatherdata(filename='step_', steps=10, output='total')
    current_experiment.importnumericaldatas()
61  current_experiment.plot()
```

Here are some explanations:

- Load several librairies, several are available on the internet, others were created or modified.
- Defines some basic parameters: temperature of the injected CO_2, the number of time step concerning the discretization, and the total time of a run.
- defines a step, with two different cases: the first step (step0) which use initial datas, and all other steps which use the previous ones. It defines all the parameters, run the simulation, export the output datas and save them.
- It enables to concatenate the different output file
- It uses the experiment python class in order to plot the simulation data versus the experimental ones.

Annexe C: Surface thermocouples documentation

I write a document to Mario Magliocco in order to explain why we put surface thermocouples, which one we chose, how the are settle, where there are settle and how they are import in the LabView (Microsoft) Software file. I also enclosed the user's guide and documentation of those hardware, given by the company Omega.

I Why we want to put surface thermocouples on the vessel wall ?

We want to get boundary conditions for the experiments, and to be able to see if there is some heat exchanges on the vessel wall. It also enables us to establish a more accurate radial temperature profile.

II What are those thermocouples

II.1 Thermocouples

Thermocouple used are Surface Thermocouple with Self-Adhesive Backing from Omega. There are T-thermocouples : Copper Constanstan. The general specification are on the Omega Website (link), or at the end of this document. Those thermocouples can work in the following temperature range : from -60°Celsius to 175°Celcius (from -75°F to 350°F).

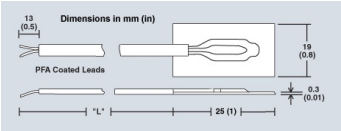


Figure 1 – Thermocouple scheme

II.2 Connectors

Connectors used are Miniature Thermocouple Connectors from Omega. They are male and female connectors, suited for T-wires. The general specifications are on the Omega website(link), or included at the end of this document.

II.3 Wires

Wires used are TG-T-30 from Omega. Other wire is used (orange wrapped one). Omega website(link), or included at the end of this document.

III How they are set ?

III.1 Where we want to put them ?

In order to choose the location on the vessel, we can look at Mario Magliocco results :

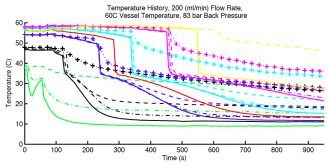


Figure 2 – Temperature in the vessel vs time data

If we look for the initial conditions, the temperature gradient versus vertical location appears more important at the bottom of the vessel. As a consequence, we will try to put surface thermocouples all along the vessel - and on both the top and bottom cap - but more focused at the bottom of the vessel. Naturally, we also want to collect data at the same vertical levels than the thermocouples inside the vessel.

III.2 General settings

The surface thermocouples are set on a cleaned part of the vessel : Then the wires come

Figure 3 – Temperature in the vessel vs time data

through a PVC pipe, and are connected to other wires thanks to male/female connectors. The datas are then received by the computer.

III.3 precise location and channels used :

These informations are essential in order to import the datas :

Thermocouples	Target location	Real location	M Conn.	F Conn.	Channel	Input
A	0	0	A	A	5	25
B	5	5.3	B	B	7	27
C	10	9.7	C	C	8	28
D	15	14.7	D	D	9	29
E	20	19.8	E	E	10	30
F	25	24.6	F	•	•	•
G	30	29.6	G	G	11	31
H	40	39.5	H	H	12	32
I	50.8	50.5	I	I	13	33
J	top cap	58.7	J	J	6	26

Location are give in centimeters. Connector letters and channels are not suited because I cut some wires during the settings.

III.4 Remarks :

- Wires from thermocouples to female connectors are really weak. In order to prevent them fro being cut, they are put in a PVC tube from the vessel to the wall.
- As some cable were missing, the thermocouples F is not linked to the computer. This thermocouple seem to be the less usefull one, as we want to focus on both the inside thermocouples vertical positions and the bottom of the vessel. However, this thermocouples have been tested with a female connector and a channel, it works.
- The simulation models, run with python, should use the initial datas from those thermocouples in order to use boundary conditions.

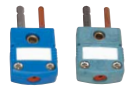
Most Popular Miniature Connectors

Type SMPW Glass Filled Nylon
-29 to 220°C (-20 to 425°F) Service Temperature
Type HMPW Liquid Crystal Polymer
260°C (500°F) Service Temperature

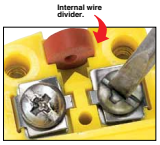
MONOGRAM SERIES

HMPW/SMPW Basic Pair

- Heavy Duty Construction
- Color-Coded for ANSI and IEC
- Captive Cover Screws
- Accepts Stranded or Solid Wire up to Size 20 AWG
- Quick Connect Contact Washers
- Internal Wire Divider
- Combination Phillips/Slot Screws
- Type HMPW Environmentally Friendly Cadmium-Free

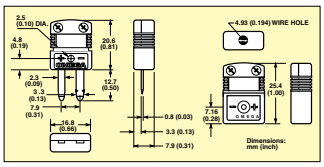


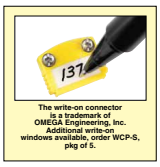
SMPW — Glass-Filled Nylon
HMPW — Cadmium Free Liquid Crystal Polymer
Shown smaller than actual size.



Visit omega.com to See OMEGA's Complete Selection of Connector Accessories

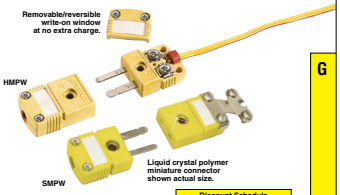
We make running changes when technical advances allow. Check at time of ordering for additional features.





The write-on connector is a trademark of OMEGA Engineering, Inc. Additional write-on windows available, order WCP-S, pkg of 5.

Quick connect contact washers. Additional washers available, order CW-SMP-20, pkg of 20, or CW-SMP-100, pkg of 100



Also Available!

Transmitters and Ferrite Cores for EMI Protection

STCTX built-in 4 to 20 mA transmitter.

PATENTED

PCLM-SMP-FT cable clamp with ferrite core for EMI protection. Connector sold separately.

Order additional silicone wire grommets! SR-SMP-10, pkg of 10 or SR-SMP-100, pkg of 100

Discount Schedule	
1-10	Net
11-49	5%
50-99	10%
100-999	15%
1000-4999	20%
5000 and over	25%

To Order *Visit omega.com/smpw_smp_hmp_hmpw for Pricing and Details*

Alloy Code†	Compensating Alloy Used in Connector +	—	Shell Color	Glass-Filled Nylon Model Number w/Window	Liquid Crystal Polymer Model Number w/Window
K	Chromega®	Alomega®	Yellow	SMPW-K-(*)	HMPW-K-(*)
T	Copper	Constantan	Blue	SMPW-T-(*)	HMPW-T-(*)
J	Iron	Constantan	Black	SMPW-J-(*)	HMPW-J-(*)
E	Chromega®	Constantan	Purple	SMPW-E-(*)	HMPW-E-(*)
R/S	Copper	RNX/SNX	Green	SMPW-R/S-(*)	HMPW-R/S-(*)
G	GPX	GNX	Red/Grn	SMPW-G-(*)	HMPW-G-(*)
C	CPX	CNX	Red	SMPW-C-(*)	HMPW-C-(*)
D	DPX	DNX	Red/Wht	SMPW-D-(*)	HMPW-D-(*)
U	Copper	Copper	White	SMPW-U-(*)	HMPW-U-(*)
N	Omega-P®	Omega-N®	Orange	SMPW-N-(*)	HMPW-N-(*)

**To Order: Specify "MF" for a connector pair, "M" for male connector only, or "F" for female connector only.*
†J, K, T, E, R, S and B are ANSI designations. OMEGALOY® is generically known as Nicrosil-Nisil. ANSI color coded models shown.
***Note:** Non-Window models available, please consult Sales for details.*
Type U (uncompensated) connectors are used with Type B thermocouples (Pt/6%Rh-Pt/30%Rh).
***Ordering Examples: SMPW-K-MF,** glass-filled nylon, Type K connector pair. **SMPW-K-M,** glass-filled nylon, Type K male connector. **SMPW-K-F,** glass-filled nylon, Type K female connector. **HMPW-K-MF,** liquid crystal polymer, Type K connector pair.*

T Duplex Insulated Copper-Constantan Duplex ANSI Type T

"SLE" Special Limits of Error Available

MADE IN USA

ANSI color code shown | To order IEC color code visit omega.com

ANSI Color Code: Positive Wire, Blue; Negative Wire, Red; Overall, Brown
OMEGA Engineering does not use reprocessed PFA or PVC in manufacturing thermocouple wire.

To order *visit omega.com/gg_t_tc_wire for Pricing and Details*

Insulation	AWG No.	Model Number	Type Wire	Insulation: Conductor	Insulation: Overall	Max Temp °C	Max Temp °F	Nominal Size mm (inches)	Wt.† kg/300 m (lb/1000')
Glass	20	GG-T-20	Solid	Glass Braid	Glass Braid	260	500	1.5 x 2.4 (0.060 x 0.095)	4 (9)
	20	GG-T-20S	7 x 28	Glass Braid		260	500	1.5 x 2.5 (0.060 x 0.100)	4 (9)
	24	GG-T-24	Solid	Glass Braid		200	400	1.3 x 2.0 (0.050 x 0.080)	3 (5)
	24	GG-T-24S	7 x 32	Glass Braid		200	400	1.3 x 2.2 (0.050 x 0.085)	3 (5)
	26	GG-T-26	Solid	Glass Braid		200	400	1.1 x 1.9 (0.045 x 0.075)	2 (4)
	28	GG-T-28	Solid	Glass Wrap		200	400	1.0 x 1.4 (0.040 x 0.055)	2 (3)
	30	GG-T-30	Solid	Glass Wrap		150	300	0.9 x 1.3 (0.037 x 0.050)	2 (3)
Glass with Stainless Steel Overbraid	20	GG-T-20-SB	Solid	Glass	SS Braid Over Glass	260	500	2.2 x 3.0 (0.090 x 0.120)	6 (14)
	20S	GG-T-20S-SB	Solid			260	500	2.2 x 3.0 (0.085 x 0.117)	5 (11)
	24	GG-T-24-SB	Solid			200	400	2.2 x 3.0 (0.085 x 0.117)	5 (11)
	24S	GG-T-24S-SB	Solid			200	400	2.2 x 3.0 (0.085 x 0.117)	5 (11)
Kapton Polymide Tape	20	KK-T-20	Solid	Fused Polymide Tape	Fused Polymide Tape	260	500	1.5 x 2.5 (0.060 x 0.100)	5 (11)
	24	KK-T-24	Solid			260	500	1.3 x 1.9 (0.050 x 0.075)	3 (6)
	30	KK-T-30	Solid			260	500	1.0 x 1.4 (0.040 x 0.055)	3 (5)
PFA Glass	30	TG-T-30	Solid	PFA	Glass Braid	150	300	0.9 x 1.2 (0.034 x 0.047)	1 (2)
	36	TG-T-36	Solid			150	300	0.7 x 1.0 (0.028 x 0.038)	1 (2)
	40	TG-T-40	Solid			150	300	0.7 x 0.9 (0.026 x 0.035)	1 (2)
Neoflon PFA (High Performance)	20	TT-T-20	Solid	PFA	PFA	260	500	1.7 x 3.0 (0.068 x 0.116)	5 (11)
	20	TT-T-20S	7 x 28			260	500	1.9 x 3.2 (0.073 x 0.126)	5 (11)
	22	TT-T-22S	7 x 30			260	500	1.7 x 3.4 (0.065 x 0.133)	4 (9)
	24	TT-T-24	Solid			200	400	1.4 x 2.4 (0.056 x 0.092)	3 (7)
	24	TT-T-24S	7 x 32			200	400	1.6 x 2.6 (0.063 x 0.102)	3 (7)
	30	TT-T-30††	Solid			150	300	0.6 x 1.0 (0.024 x 0.040)	1 (2)
	36	TT-T-36††	Solid			150	300	0.5 x 0.8 (0.019 x 0.030)	1 (2)
	40	TT-T-40††	Solid			150	300	0.4 x 0.7 (0.017 x 0.026)	1 (2)
PFA Polymer w/Twisted and Shielded Conductors	20	TT-T-20-TWSH	Solid	PFA Polymer	PFA Polymer and Shielding	260	500	3.7 (0.15)	9 (20)
	20S	TT-T-20S-TWSH	7 x 28			260	500	3.8 (0.15)	9 (20)
	24	TT-T-24-TWSH	Solid			260	500	2.7 (0.11)	4 (9)
	24S	TT-T-24S-TWSH	7 x 32			260	500	2.9 (0.12)	4 (9)
Neoflon FEP	20	FF-T-20	Solid	FEP	FEP	200	392	1.7 x 3.0 (0.068 x 0.116)	5 (11)
	24	FF-T-24	Solid			200	392	1.4 x 2.4 (0.056 x 0.092)	3 (7)
FEP Polymer w/Twisted and Shielded Conductors	20	FF-T-20-TWSH	Solid	FEP Polymer	FEP Polymer and Shielding	200	392	3.7 (0.15)	9 (20)
	20S	FF-T-20S-TWSH	7 x 28			200	392	3.8 (0.15)	9 (20)
	24	FF-T-24-TWSH	Solid			200	392	2.7 (0.11)	4 (9)
	24S	FF-T-24S-TWSH	7 x 32			200	392	2.9 (0.12)	4 (9)
TFE Tape Polymer	20	TFE-T-20	Solid	TFE Tape Polymer	Fused TFE Tape Polymer	260	500	1.5 x 2.5 (0.060 x 0.100)	5 (11)
	20S	TFE-T-20S	7 x 28			260	500	1.5 x 2.7 (0.060 x 0.105)	5 (11)
	24	TFE-T-24	Solid			260	500	1.3 x 1.9 (0.050 x 0.075)	3 (6)
	24S	TFE-T-24S	7 x 32			260	500	1.3 x 2.2 (0.050 x 0.085)	3 (6)
Polyvinyl	24	PP-T-24	Solid	Polyvinyl	Polyvinyl	105	221	1.9 x 3 (0.075 x 0.120)	5 (10)
	24	PP-T-24S	7 x 32		Polyvinyl (Rip Cord)**	105	221	1.9 x 3.1 (0.080 x 0.130)	5 (10)
	24	PR-T-24	Solid			105	221	1.3 x 2.2 (0.050 x 0.086)	3 (5)

See Fused Tape Insulated TFE-T Series.
*** Two insulated leads bonded together, but with no overjacket.*
† Weight of spool and wire rounded to the next highest lb. (does not include packing material).
†† Overall color clear, ††† To order special limits of error wire, add "-SLE" to model number before spool length.
***Ordering Example: TT-T-24-SLE-1000,** 1000' (300 m) of Type T duplex insulated special limits of error thermocouple wire.*

Discount Schedule (1000' spools only)	
3 to 4 spools	10%
5 to 9 spools	15%
10 to 19 spools	20%

Note: Published prices are based on market value at time of printing and are subject to change due to Nickel surcharges, Chromium and precious-metal market fluctuations.