Thermal Energy Storage Simulation

Fundamentals of CFD Methods

L.E

December 2020

Computational Science and Engineering MSc

Contents

1 Introduction

This project is part of the course 151-0182-00L Fundamentals of CFD Methods. It consists of the derivation, implementation and testing of numerical schemes in order to solve the model equations for a thermal energy storage in one spatial dimension. The model consists of a hot fluid (advection diffusion equation) passing through a solid (diffusion equation).

We perform the derivation of the FTBS scheme using the finite volume method. Only the derivation for the uncoupled equations is performed in this report.

We then implement the solver in C++ from scratch and use Python in order to get the necessary plots for validation and testing.

2 Numerical Methods

2.1 Introduction

The project involves solving the following coupled PDEs.

$$\begin{aligned} \varepsilon\rho_f C_{p,f}\frac{\partial T_f}{\partial t} + \varepsilon\rho_f C_{p,f}u_{f,i}\frac{\partial T_f}{\partial x} &= k_f\frac{\partial^2 T_f}{\partial x^2} + h_v\left(T_s - T_f\right) \\ (1-\varepsilon)\rho_s C_s\frac{\partial T_s}{\partial t} &= k_s\frac{\partial^2 T_s}{\partial x^2} - h_v\left(T_s - T_f\right) \end{aligned} \tag{1}$$

rewriting yields

$$\begin{aligned} \frac{\partial T_f}{\partial t} + u_{f,i}\frac{\partial T_f}{\partial x} &= \alpha_f\frac{\partial^2 T_f}{\partial x^2} + h_{v,f}\left(T_s - T_f\right) \\ \frac{\partial T_s}{\partial t} &= \alpha_s\frac{\partial^2 T_s}{\partial x^2} - h_{v,s}\left(T_s - T_f\right) \end{aligned} \tag{2}$$

with

$$\alpha_f = \frac{k_f}{\epsilon\rho_f C_{p,f}} \tag{3}$$

$$\alpha_s = \frac{k_s}{(1-\epsilon)\rho_s C_s} \tag{4}$$

We first start by solving the non-coupled equations i.e with $h_v = 0$. We then build upon the uncoupled equation solver in order to solve the coupled equations and make meaningful simulations.

2.2 Non-Coupled Fluid Advection-Diffusion Equation

$$\frac{\partial T_f}{\partial t} + u_{f,i}\frac{\partial T_f}{\partial x} = \alpha_f\frac{\partial^2 T_f}{\partial x^2}\ ,\ T_f \in C^2(\bar{\Omega})\ ,\ \Omega =]-0.5, (N-1)h+0.5[\times]0,\infty[\tag{5}$$

with $\frac{\partial T_f}{\partial x} = 0$ on $\partial\Omega$ and $T_f(0,t) = T_L$.

In order to derive the finite volume method we need to integrate over $\Omega_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}] \subset \Omega$

$$\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \frac{\partial T_f}{\partial t} dx + \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} u_{f,i} \frac{\partial T_f}{\partial x} dx = \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \alpha_f \frac{\partial^2 T_f}{\partial x^2} dx \tag{6}$$

this yields

$$h \frac{d\bar{T}_{f,i}}{dt} + u_{f,i}(T_{f,i+\frac{1}{2}} - T_{f,i-\frac{1}{2}}) = \alpha_f (\frac{\partial T_{f,i+\frac{1}{2}}}{\partial x} - \frac{\partial T_{f,i-\frac{1}{2}}}{\partial x}) \tag{7}$$

we now want to reconstruct the face values with :

$$T_{i\pm\frac{1}{2}} \approx \sum_{j=l}^{r} \omega_j \bar{T}_{i\pm\frac{1}{2}+j} \quad and \quad \frac{\partial T}{\partial x}\Big|_{x_{i\pm\frac{1}{2}}} \approx \sum_{k=d}^{m} \omega_k \bar{T}_{i\pm\frac{1}{2}+k} \tag{8}$$

for this method we take $l = d = r = -\frac{1}{2}$ and $m = \frac{1}{2}$.

Partial Derivatives on the face :

For the first partial derivative we get by (4) :

$$\frac{\partial T}{\partial x}\Big|_{x_{i+\frac{1}{2}}} \approx \omega_{-\frac{1}{2}} \bar{T}_i + \omega_{\frac{1}{2}} \bar{T}_{i+1} \tag{9}$$

Recall the following relationship between cell average and centroid value :

$$\bar{T}_i = T_i + \frac{h^2}{24} \frac{\partial^2 T}{\partial x^2}\Big|_{x_i} + O(h^4) \tag{10}$$

by applying (6) at the first order we get,

$$\frac{\partial T}{\partial x}\Big|_{x_{i+\frac{1}{2}}} \approx \omega_{-\frac{1}{2}} T_i + \omega_{\frac{1}{2}} T_{i+1} \tag{11}$$

We can continue by developing T_i and T_{i+1} as taylor series in $x_0 = x_{i+\frac{1}{2}} = x_i + \frac{h}{2}$.

$$T_i = T_{i+\frac{1}{2}} - \frac{h}{2} \frac{\partial T}{\partial x}\Big|_{x_{i+\frac{1}{2}}} + O(h^2) \tag{12}$$

$$T_{i+1} = T_{i+\frac{1}{2}} + \frac{h}{2} \frac{\partial T}{\partial x}\Big|_{x_{i+\frac{1}{2}}} + O(h^2) \tag{13}$$

plugging back (8) and (9) in our expression we get:

$$\frac{\partial T}{\partial x}\Big|_{x_{i+\frac{1}{2}}} \approx (\omega_{-\frac{1}{2}} + \omega_{\frac{1}{2}}) T_{i+\frac{1}{2}} + (\omega_{\frac{1}{2}} - \omega_{-\frac{1}{2}}) \frac{h}{2} \frac{\partial T}{\partial x}\Big|_{x_{i+\frac{1}{2}}} \tag{14}$$

solving for $\omega_{-\frac{1}{2}}$ and $\omega_{\frac{1}{2}}$ we get $\omega_{\frac{1}{2}} = \frac{1}{h}$ and $\omega_{-\frac{1}{2}} = -\frac{1}{h}$.

thus by applying the same process for the other face value we end up with:

$$\frac{\partial T}{\partial x}\Big|_{x_{i+\frac{1}{2}}} \approx \frac{1}{h}(\bar{T}_{i+1} - \bar{T}_i) \tag{15}$$

$$\frac{\partial T}{\partial x}\Big|_{x_{i-\frac{1}{2}}} \approx \frac{1}{h}(\bar{T}_i - \bar{T}_{i-1}) \tag{16}$$

which gives :

$$\frac{\partial T_{f,i+\frac{1}{2}}}{\partial x} - \frac{\partial T_{f,i-\frac{1}{2}}}{\partial x} = \frac{1}{h}(\bar{T}^n_{f,i+1} - 2\bar{T}^n_{f,i} + \bar{T}^n_{f,i-1}) \tag{17}$$

Face values :

For the face values we get by applying successively (6) and (8) at the smallest order:

$$T_{i+\frac{1}{2}} \approx \omega_{-\frac{1}{2}}\bar{T}_i \approx \omega_{-\frac{1}{2}}T_i \approx \omega_{-\frac{1}{2}}T_{i+\frac{1}{2}} \tag{18}$$

identically we have

$$T_{i-\frac{1}{2}} \approx \omega_{-\frac{1}{2}}T_{i-1} \approx \omega_{-\frac{1}{2}}T_{i-\frac{1}{2}} \tag{19}$$

It follows that both $\omega_{-\frac{1}{2}} = 1$ and therefore we get the approximation :

$$(T_{f,i+\frac{1}{2}} - T_{f,i-\frac{1}{2}}) = \bar{T}_{f,i} - \bar{T}_{f,i-1} \tag{20}$$

Time derivative :

Using a forward Euler scheme for the time derivative we get :

$$\bar{T}^{n+1}_{f,i} = \bar{T}^n_{f,i} + \Delta t F(\bar{T}^n_{f,i+1}, \bar{T}^n_{f,i}, \bar{T}^n_{f,i+1}) \tag{21}$$

The fully discrete equation is therefore given by :

$$\bar{T}^{n+1}_{f,i} = \bar{T}^n_{f,i} - \frac{u_{f,i}\Delta t}{\Delta x}(\bar{T}^n_{f,i} - \bar{T}^n_{f,i-1}) + \frac{\alpha_f \Delta t}{(\Delta x)^2}(\bar{T}^n_{f,i+1} - 2\bar{T}^n_{f,i} + \bar{T}^n_{f,i-1}) \tag{22}$$

Boundary Conditions :

From the two Neumann boundary conditions and the left Dirichlet boundary condition we derive the following equation for the left boundary :

$$\bar{T}^{n+1}_{f,1} = \bar{T}^n_{f,1} - \frac{u_{f,1}\Delta t}{\Delta x}(\bar{T}^n_{f,1} - T_L) + \frac{\alpha_f \Delta t}{(\Delta x)^2}(\bar{T}^n_{f,2} - \bar{T}^n_{f,1}) \tag{23}$$

and on the right :

$$\bar{T}_{f,N}^{n+1} = \bar{T}_{f,N}^{n} - \frac{u_{f,N}\Delta t}{\Delta x}(\bar{T}_{f,N}^{n} - \bar{T}_{f,N-1}^{n}) + \frac{\alpha_f \Delta t}{(\Delta x)^2}(\bar{T}_{f,N-1}^{n} - \bar{T}_{f,N}^{n}) \tag{24}$$

MMS mode :
Using the method of manufactured solution with $T_f(x) = cos(kx)$ we get :

$$\begin{aligned}\bar{T}_{f,i}^{n+1} = \bar{T}_{f,i}^{n} - \frac{u_{f,i}\Delta t}{\Delta x}(\bar{T}_{f,i}^{n} - \bar{T}_{f,i-1}^{n}) + \frac{\alpha_f \Delta t}{(\Delta x)^2}(\bar{T}_{f,i+1}^{n} - 2\bar{T}_{f,i}^{n} + \bar{T}_{f,i-1}^{n}) \\ +\Delta t(\alpha_f k^2 cos(kx_i) - u_{f,i} k sin(kx_i))\end{aligned} \tag{25}$$

2.3 Non-Coupled Solid Diffusion Equation

$$\frac{\partial T_s}{\partial t} = \alpha_s \frac{\partial^2 T_s}{\partial x^2} \ , \ T_s \in C^2(\bar{\Omega}) \ , \ \Omega =]-0.5, (N-1)h + 0.5[\times]0, \infty[\tag{26}$$

with $\frac{\partial T_s}{\partial x} = 0$ on $\partial\Omega$.

We use the same discretization for the second order partial derivative and a forward Euler scheme for the time derivative as in the fluid equation.

The fully discrete equation is therefore given by :

$$\bar{T}_{s,i}^{n+1} = \bar{T}_{s,i}^{n} + \frac{\alpha_s \Delta t}{(\Delta x)^2}(\bar{T}_{s,i+1}^{n} - 2\bar{T}_{s,i}^{n} + \bar{T}_{s,i-1}^{n}) \tag{27}$$

Boundary Conditions :

Applying Neumann boundary condition on the boundary we get :

$$\bar{T}_{s,1}^{n+1} = \bar{T}_{s,1}^{n} + \frac{\alpha_s \Delta t}{(\Delta x)^2}(\bar{T}_{s,2}^{n} - \bar{T}_{s,1}^{n}) \tag{28}$$

and

$$\bar{T}_{s,N}^{n+1} = \bar{T}_{s,N}^{n} + \frac{\alpha_s \Delta t}{(\Delta x)^2}(\bar{T}_{s,N-1}^{n} - \bar{T}_{s,N}^{n}) \tag{29}$$

MMS mode :
Using the method of manufactured solution with $T_s(x) = cos(kx)$ we get :

$$\bar{T}_{s,i}^{n+1} = \bar{T}_{s,i}^{n} + \frac{\alpha_s \Delta t}{(\Delta x)^2}(\bar{T}_{s,i+1}^{n} - 2\bar{T}_{s,i}^{n} + \bar{T}_{s,i-1}^{n}) + \Delta t \alpha_s k^2 cos(kx_i) \tag{30}$$

2.4 Stability conditions

From Von-Neumann analysis the following stability conditions can be derived for the non coupled system of equation.

$$\sigma^2 \leq \sigma + 2d_f \leq 1 \tag{31}$$

$$2d_s \leq 1 \tag{32}$$

where

$$\begin{aligned} \sigma &= \frac{u_{f,i}\Delta t}{\Delta x} \\ d_f &= \frac{\alpha_f \Delta t}{(\Delta x)^2} \\ d_s &= \frac{\alpha_s \Delta t}{(\Delta x)^2} \end{aligned} \tag{33}$$

2.5 Coupled Equations

$$\frac{\partial T_f}{\partial t} + u_{f,i}\frac{\partial T_f}{\partial x} = \alpha_f \frac{\partial^2 T_f}{\partial x^2} - h_{v,f}(T_f - T_s) \tag{34}$$

$$\frac{\partial T_s}{\partial t} = \alpha_s \frac{\partial^2 T_s}{\partial x^2} - h_{v,s}(T_s - T_f) \tag{35}$$

The two step numerical method used is the following :

$$\begin{aligned} \bar{T}^*_{f,i} = \bar{T}^n_{f,i} - \frac{u_{f,i}\Delta t}{\Delta x}\left(\bar{T}^n_{f,i} - \bar{T}^n_{f,i-1}\right) + \frac{\alpha_f \Delta t}{(\Delta x)^2}\left(\bar{T}^n_{f,i+1} - 2\bar{T}^n_{f,i} + \bar{T}^n_{f,i-1}\right), \\ \bar{T}^*_{s,i} = \bar{T}^n_{s,i} + \frac{\alpha_s \Delta t}{(\Delta x)^2}\left(\bar{T}^n_{s,i+1} - 2\bar{T}^n_{s,i} + \bar{T}^n_{s,i-1}\right) \end{aligned} \tag{36}$$

and then solve

$$\begin{bmatrix} 1 + h_{v,f}\Delta t & -h_{v,f}\Delta t \\ -h_{v,s}\Delta t & 1 + h_{v,s}\Delta t \end{bmatrix} \begin{Bmatrix} \bar{T}^{n+1}_{f,i} \\ \bar{T}^{n+1}_{s,i} \end{Bmatrix} = \begin{Bmatrix} \bar{T}^*_{f,i} \\ \bar{T}^*_{s,i} \end{Bmatrix} \tag{37}$$

2.6 Managing the charging/discharging phases

In order to switch from charging ($u_{f,i} > 0$, FTBS scheme stable) to discharging ($u_{f,i} < 0$) we need to change our stencil for the advection term and thus the boundary condition. This is the FTFS scheme.

For the face values coming back to equation (4) and using $l = r = \frac{1}{2}$ we get by applying successively (6) and (8) at the smallest order:

$$T_{i+\frac{1}{2}} \approx \omega_{\frac{1}{2}}\bar{T}_{i+1} \approx \omega_{\frac{1}{2}} T_{i+1} \approx \omega_{\frac{1}{2}} T_{i+\frac{1}{2}} \tag{38}$$

identically we have

$$T_{i-\frac{1}{2}} \approx \omega_{\frac{1}{2}} T_i \approx \omega_{\frac{1}{2}} T_{i-\frac{1}{2}} \tag{39}$$

It follows that both $\omega_{\frac{1}{2}} = 1$ and therefore we get the approximation :

The method is therefore given by :

$$\begin{aligned} \bar{T}^{n+1}_{f,i} = \bar{T}^n_{f,i} - \frac{u_{f,i}\Delta t}{\Delta x}(\bar{T}^n_{f,i+1} - \bar{T}^n_{f,i}) + \frac{\alpha_f \Delta t}{(\Delta x)^2}(\bar{T}^n_{f,i+1} - 2\bar{T}^n_{f,i} + \bar{T}^n_{f,i-1}) \\ -h_{v,f}\Delta t(\bar{T}^n_{f,i} - \bar{T}^n_{s,i}) \end{aligned} \tag{40}$$

Boundary Conditions :

On the left boundary we get :

$$\bar{T}_{f,1}^{n+1} = \bar{T}_{f,1}^{n} - \frac{u_{f,1}\Delta t}{\Delta x}(\bar{T}_{f,2}^{n} - \bar{T}_{f,1}^{n}) + \frac{\alpha_f \Delta t}{(\Delta x)^2}(\bar{T}_{f,2}^{n} - \bar{T}_{f,1}^{n}) \tag{41}$$

and on the right :

$$\bar{T}_{f,N}^{n+1} = \bar{T}_{f,N}^{n} - \frac{u_{f,N}\Delta t}{\Delta x}(T_R - \bar{T}_{f,N}^{n}) + \frac{\alpha_f \Delta t}{(\Delta x)^2}(\bar{T}_{f,N-1}^{n} - \bar{T}_{f,N}^{n}) \tag{42}$$

3 Results and Analysis

3.1 Order Verification Study with Method of Manufactured Solution

Results :
Uncoupled fluid temperature :

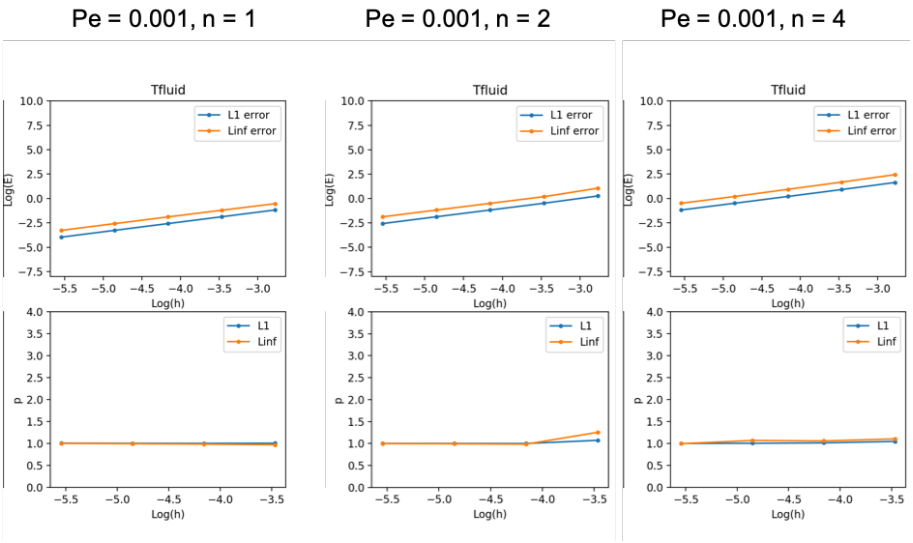

Figure 1: OVS for the fluid temperature with Pe = 0.001

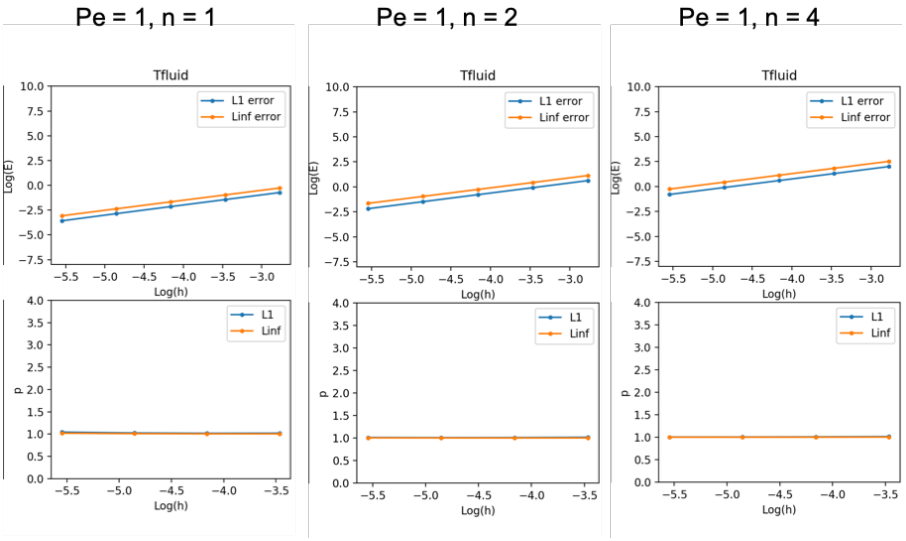

Figure 2: OVS for the fluid temperature with Pe = 1

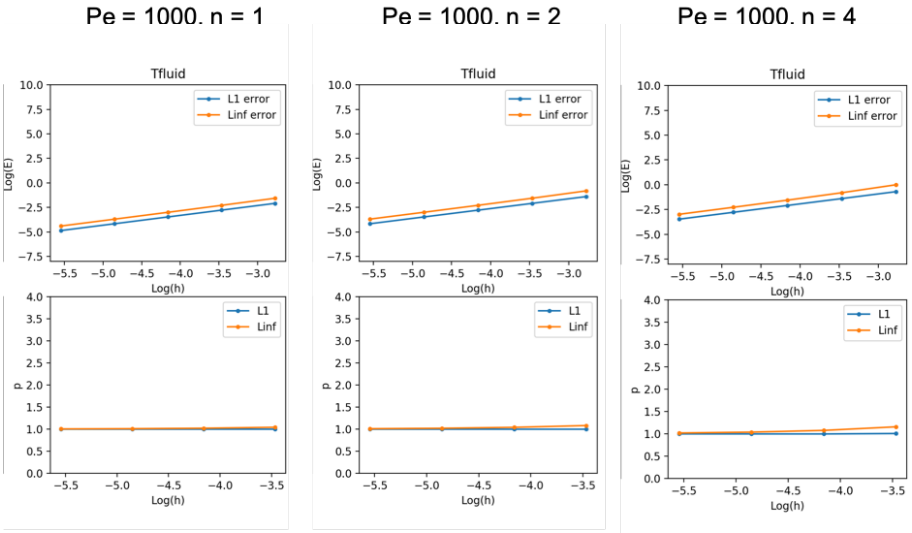

Figure 3: OVS for the fluid temperature with Pe = 1000

Uncoupled solid temperature :

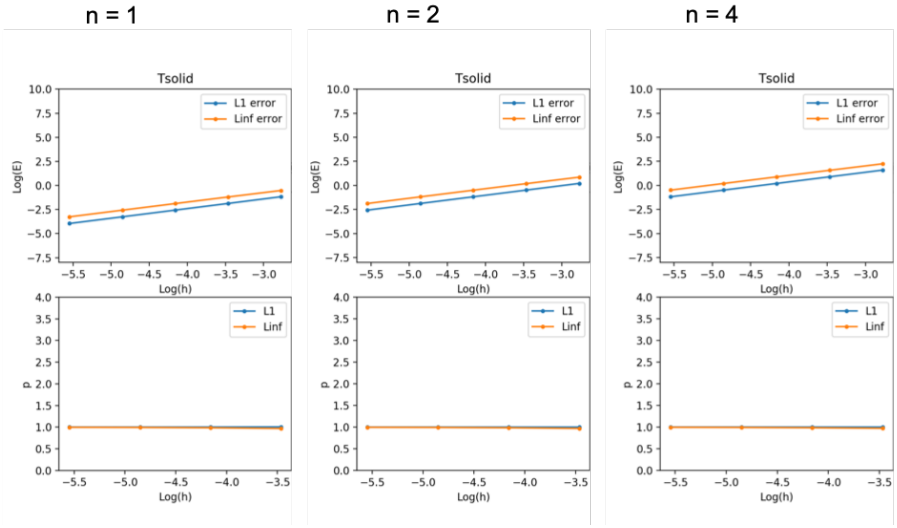

Figure 4: OVS for the solid temperature.

Coupled fluid and solid temperatures :

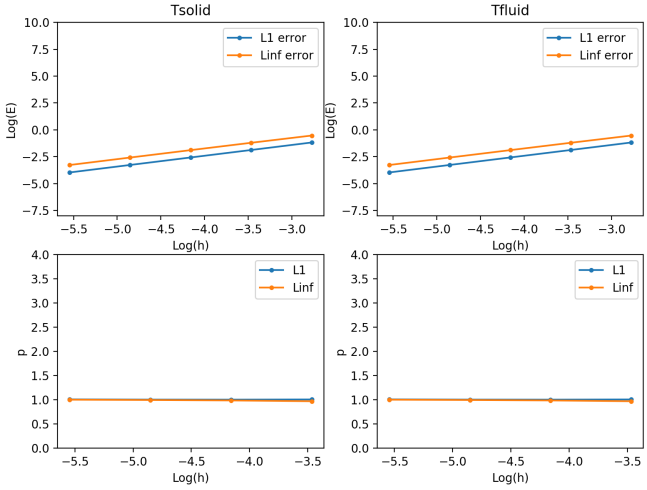

Figure 5: OVS for the coupled equations with Pe $= 0.001$ ($n_f = 2$, $n_s = 1$).

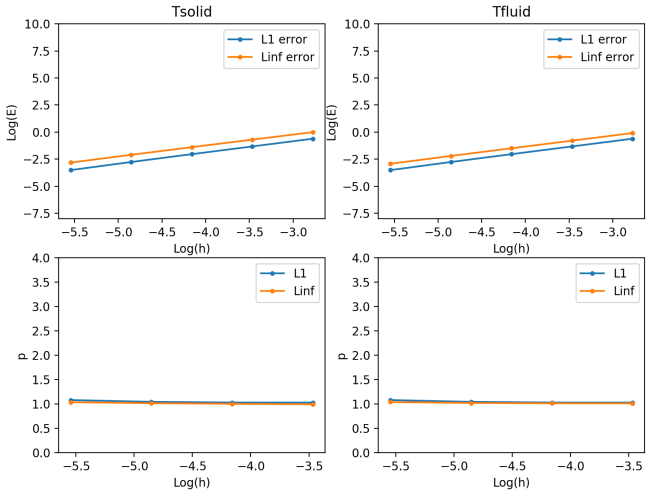

Figure 6: OVS for the coupled equations with Pe = 1 ($n_f = 2$, $n_s = 1$).

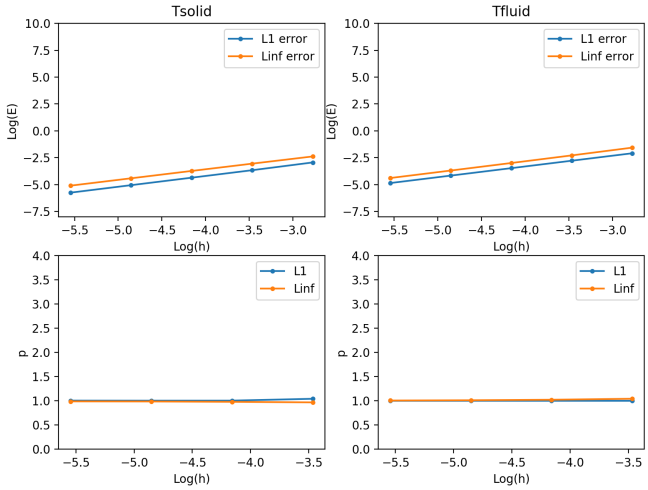

Figure 7: OVS for the coupled equations with Pe = 1000 ($n_f = 2$, $n_s = 1$).

In this last part the solid OVS are all the same as Pe doesn't affect α_s.

Analysis :
Firstly it is important to mention that we can prove that our discretization for

the advection term is of order 1 and that the approximation for the diffusion term is of order 2. Therefore the theory predict for the uncoupled equation an accuracy of 2 for the solid solution and between 1 and 2 for the fluid solution (close to two for Pe = 0.001, between 1 and 2 for Pe = 1 and 1 for Pe = 1000). This comes from the fact that Pe will change the weight on the diffusion and advection terms and this should therefore be seen on the convergence plot.
For the coupled equations the expected order is not well defined as both equation depends on the other one and on its order of accuracy. However the results are expected to be between 1 and 2.

We can see that it is not the case in the results provided above, I only get order of convergence 1.

After countless hours of debugging I am only able to tell where this problem does not come from. In order to determine if the issue was from the determination of the steady state I tried many different threshold values (the final code implements 1e-8 for the normalized norm difference between the solution at timestep n and n+1) and also to use just a fixed number of timestep without threshold and always got similar results (although threshold equal or higher to 1e-6 seems to give really inconsistent results with respect to the number of cells most likely showing that the steady state was not reached).
Secondly the implementation of the calculation for the next timestep were checked so many times that it is highly unlikely that the problem comes from a bad translation of the mathematical discrete solution, to check this I also implemented a fourth order finite difference scheme of the diffusion term and did not see any change in the OVS (available in the github history of the project). Seeing also the comparison with analytical solution (3.2) seems to show that overall the implemented solution approximates the right problem.
This behaviour could come from the way I calculate the error but my functions L1Error()) and LinfError() were thoroughly tested with known values and they always agree with the theory i.e the L_1 norm is bounded from above by the L_∞ norm. I also did not see any problem in the way I refine my grid (number of cells multiplied by 2 at each refinement).
The implementation of the plotting was also carefully checked and the right logarithm base ($log_{\frac{1}{2}}(x)$) was used for calculating the slope, in addition it is easy to see when doing the simulation that the error is always multiplied by $\frac{1}{2}$ for each refinement.
The grid refinement is implemented from 8 cells up to 256 cells for speed however testing up to 1024 cells did not change the rate.

To conclude, my code has a mistake as it should not behave like this, however it seems to be relatively fast and to approximate the real solutions quite well.

3.2 Comparison with Exact Solution

In order to test our numerical solution we can plot it against a given analytic solution using the same parameters.

Results :

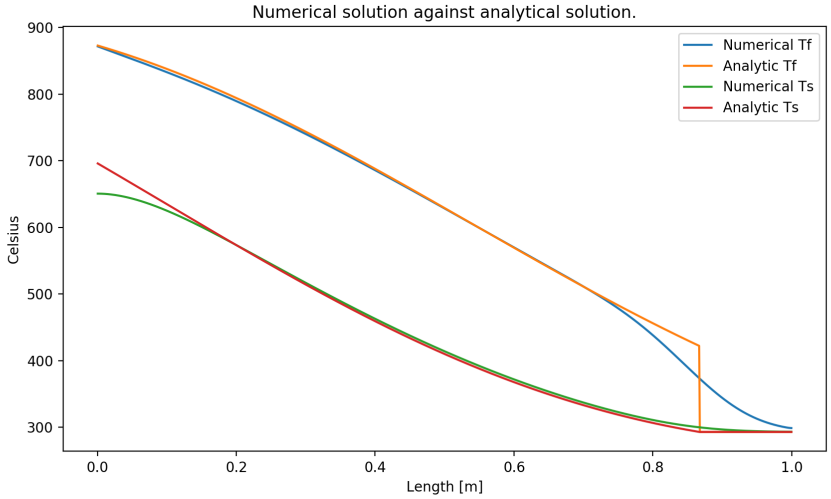

Figure 8: Plot of the analytic solutions against the numerical solutions using the parameters given in assignment 5 (see *setuppr5.txt* for program input). T_f denotes the fluid temperature and T_s the solid's.

Analysis : We can see from this plot that the numerical solution seems to be overall consistent with the analytic solution. We can however see that at $x \simeq 0.85$ the numerical solution does not do a good job at representing the sudden change in temperature of the liquid phase at the front of the wave. There is also and apparent diffusion error for the solid solution at the left of the graph.

3.3 Storage Design Study

Here are the results of the storage design study for different diameters.

Results :

Diameter [m]	ΔT [°C]	Exergy Efficiency	Capacity Factor
4.0	103.055	0.950	0.399
5.0	114.964	0.938	0.397
6.0	125.867	0.926	0.395
7.0	136.093	0.914	0.391
8.0	145.828	0.902	0.387

Obtained after 30 cycles.

Analysis :
Here the most noticeable pattern is the fact that both the exergy efficiency and capacity factor are diminishing with increasing well diameter. This means that by increasing the diameter of the container we diminish the amount of energy we are able to extract for the same input.

It is however the opposite for ΔT. The bigger the diameter, the hotter the right side of the well gets. This is due to the fact that the diameter is involved in the calculation of the height of the well of constant volume and of u_f, $h_{v,s}$ and $h_{v,f}$. Therefore the temperature profile is not the same.

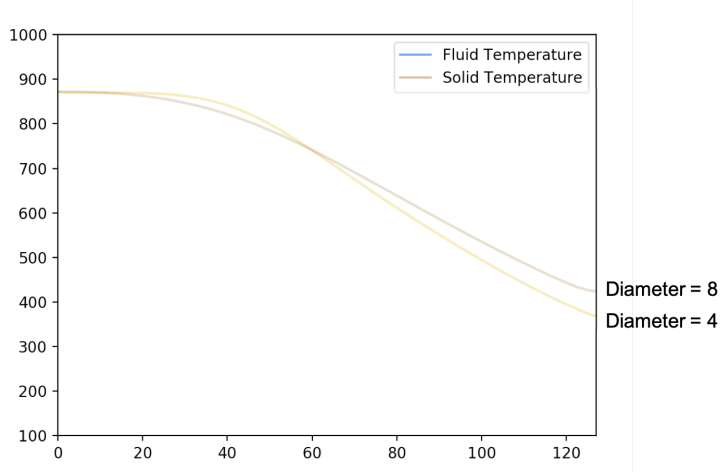

Figure 9: Plot of the end of charging state temperature profile for diameter = 8 and 4.)

Overall I think the model used was a good approximation of a real life scenario as it is taking advantage of the rotational symmetry of a cylindrical well in order to simplify to a 1D space problem. However in real life the input and output ducts are probably not as wide as the well, this issue would require to model the problem with one additional space dimension. Also the model could be improved by taking into account the potential heat losses of the well.

Regarding the numerical method used I think the most important improvement could be to use a less restrictive and higher order time integration method. The conditions on the timestep of explicit Euler method made the OVS complicated to manage and using an implicit scheme could have solved this issue (of course at the computational cost of solving an additional system of equations). I think using an implicit or even an explicit Runge-Kutta method would be a good improvement.
I think that in order to balance the added complexity of using an implicit time stepping method we could use higher order space approximations in order to work with a lower number of cells and still keep a low error.

4 Discussion

The most important conclusion I will take out from this project is that writing accurate, efficient and faultless numerical analysis code is hard and requires to be extremely careful when coding. Mistakes are hard to find mostly due to the fact that the results that can help debugging are usually far away conceptually from a single line of code as they are the result of thousands to millions of calculations. I think the most important thing is to be extremely careful when writing the code in order to minimize the time spent debugging which seems to scale exponentially with the time spent coding.

The issue that I have with my OVS made me think multiple times of just starting from a blank file again and rewriting everything carefully, however due to a lack of time I had to keep going forward in order to deliver a finished project. Therefore I think that if I had to redo the project I would focus a lot more on making the code as simple as possible and trying to get everything right the first time by reconsidering every new line of code.

5 Code Description

The code is available on GitHub :
`https://github.com/Leqr/ThermoSim`

In order to run the program you need a recent version of C++, CMake and Python 3.x with numpy and matplotlib.

The program is compiled with CMake and has the following file hierarchy :

```
Project
├── CMakeLists.txt
├── build
└── src
    └── plotter.py
```

```
├── main.cpp
├── Exporter.cpp
├── Exporter.hpp
├── Simulator.cpp
├── Simulator.hpp
├── setup.txt
├── setuppr5.txt
├── setupOVS.txt
├── setupMMS.txt
└── sol-exact-5.00000E+03.dat
```

In order to run the program the following need to be done :
1. Open a terminal and go inside the build folder (**mkdir build** then **cd build** if the folder is not present).
2. cmake ..
3. make

Now the executable **CFDSim** is built. The program can be run in different modes by passing a different argument (integer) when calling the executable. The C++ code calls python for plotting automatically, only the integer that specify the mode needs to be changed :

./CFDSim 0 runs the program in MMS mode by reaching the steady state of the MMS problem with $n = 1$ and plots the cosine against the numerical solution.

./CFDSim 1 runs the uncoupled OVS and plots the convergence graphs, by default $Pe = 1000$ and $n = 1$, these values can be changed in line 142 of *main.cpp*.

./CFDSim 2 runs the coupled OVS and plots the convergence graphs,by default $Pe = 1000$, this value can be changed in line 150 of *main.cpp*.

./CFDSim 3 compares the numerical solution with the analytical result by fetching the analytic solution in *sol-exact-5.00000E+03.dat*.

./CFDSim 4 perform the design study with diameter = 8 (can be changed in *setup.txt*, needs also to change the values of u_f, height, $h_{v,s}$ and $h_{v,f}$ with the one corresponding to the wanted diameter written at the bottom of the file) and prints the calculated exergy factor, capacity factor and temperature values, also plots a time dependent simulation of the numerical solution.

All the parameters of the simulation are defined in the setup files which are called when launching the program. Each file is relevant for a specific simulation.

setup.txt is used for the design study (mode **4**).
setuppr5.txt for comparing the numerical and analytical solution (mode **3**).
setupOVS.txt for the OVS (mode **1** and **2**).
setupMMS.txt for comparison with the cosine (mode **0**).

Overall the *main.cpp* file performs the initialisation of a Simulator and Exporter object as well as fetching the parameters and mode of the simulation.
The Simulator is launched by the main either with SIMULATOR.SIMULATE(...) which performs a standard cycle based simulation or with SIMULATOR.OVS(...) which performs the order verification study.
The Exporter object is used as an interface between the .txt output files (which are created and stored at runtime in the build folder) and the program.