Semester Project

Numerical Simulation of a Thermocline Thermal Energy Storage

Fundamentals of CFD methods

Autumn Semester 2017

Supervised by:
Dr. Andreas Haselbacher
Institute of Energy Technology

Authors:

Sthavishtha Bhopalam Rajakumar
17-940-156

Declaration of Originality

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Numerical Simulation of a Thermocline Thermal Energy Storage

Author(s)

Sthavishtha	Bhopalam Rajakumar

Supervisor

Dr. Andreas	Haselbacher

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the *Citation etiquette* [1] information
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place and date	Signature

[1] https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagi

Contents

Acknowledgements

I am deeply indebted to Dr. Andreas Haselbacher, the course intructor for clarifying my doubts during the progress of this project, and also instilling in me a sense of interest in the area of *Computational Fluid Dynamics*. Though the course project was quite challenging, he patiently answered every mail of mine, despite being busy. I am equally grateful to my fellow classmates, with whom I got the opportunity to discuss my solutions, mistakes and other problems arising during the due course of this project. Additionally, I am thankful to Prof. Dr. Ilya Karlin (tutor) and his doctoral student Benedikt Dorschner for motivating me to take up this course. The computational resources of ETH Euler cluster in running the simulations are also deeply acknowledged.

Lastly, I thank the Almighty for helping me to successfully complete this project.

Symbols

Nomenclature

α_f	Thermal diffusivity of the fluid
α_s	Thermal diffusivity of the solid
$C_{p,f}$	Specific heat of fluid phase at constant pressure
C_s	Specific heat of solid phase
d_s	Term used in the calculation of volumetric heat transfer coefficient
ϵ	porosity
E	Exergy flux
E_3	Error, indexed as 1,2 or 3 used in calculation of order of accuracy
h_v	Volumetric heat transfer coefficient
$h_{v,f}$	Volumetric heat transfer coefficient of the fluid
$h_{v,s}$	Volumetric heat transfer coefficient of the solid
H	Height of the storage medium
k_f	Thermal conductivity of fluid phase
k_s	Thermal conductivity of solid phase
m_f	Mass flow rate of fluid phase
μ_f	Absolute viscosity of fluid phase
Pe	Peclet number
Q	Actual Thermal Energy stored
Q_{max}	Maximum Thermal Energy Stored
r	Grid refinement ratio
ρ_f	Density of the fluid phase
ρ_s	Density of the solid phase
t_c	Charging time
t_d	Discharging time
t_i	Idle time
T_0	Reference temperature
T_c	Fluid inlet temperature at charging phase
T_d	Fluid outlet temperature at discharging phase
$T_{f,0}$	Fluid initial temperature

$T_{s,0}$	Solid inlet temperature
V	Volume of the storage medium

Acronyms and Abbreviations

FTBS	Forward Time Backward Spacing
FTFS	Forward Time Forward Spacing
FVM	Finite Volume Discretization
MMS	Method of Manufactured Solutions
OVS	Order of Verification Studies

Chapter 1

Introduction

The current project deals with the numerical simulation of the operation of a thermocline thermal energy storage. For this purpose, a simplified one-dimensional two-phase flow problem has been adopted and numerically solved using explicit and point-implicit methods. To assess the credibility of the results obtained by the developed numerical code, code verification has been performed with the Method of manufactured solutions and simplified exact solutions in a step-by-step manner. After obtaining good agreement of the results as a consequence of this verification, full scale simulations have been performed to replicate the actual functioning (simplified) of a thermal energy storage. Lastly, certain parameters like capacity factor, cycle exergy efficiency and outflow increase in the fluid temperature at the end of charging period have been computed by this code to assess the performance of this thermocline storage.

Chapter 2

Code Verification Studies using MMS

2.1 Governing Differential Equations of the Mathematical Model

The governing partial differential equations for the fluid and solid phase with non-zero h_v are listed here :

$$\epsilon\rho_f C_{p,f}\frac{\partial T_f}{\partial t} + \epsilon\rho_f C_{p,f}u_{f,i}\frac{\partial T_f}{\partial x} = k_f\frac{\partial^2 T_f}{\partial x^2} + h_v(T_s - T_f) \tag{2.1}$$

$$(1-\epsilon)\rho_s C_s\frac{\partial T_s}{\partial t} = k_s\frac{\partial^2 T_s}{\partial x^2} - h_v(T_s - T_f) \tag{2.2}$$

The above equations can be rewritten as :

$$\frac{\partial T_f}{\partial t} + u_{f,i}\frac{\partial T_f}{\partial x} = \alpha_f\frac{\partial^2 T_f}{\partial x^2} + h_{v,f}(T_s - T_f) \tag{2.3}$$

$$\frac{\partial T_s}{\partial t} = \alpha_s\frac{\partial^2 T_s}{\partial x^2} - h_{v,s}(T_s - T_f) \tag{2.4}$$

As a step-by-step approach to tackle this model, initially h_v is neglected, and the discrete equations are solved using FTBS approach with Finite Volume Discretization (FVM). For the charging period (positive $u_{f,i}$), the discrete equations include :

$$\overline{T_{f,i}^{n+1}} = \overline{T_{f,i}^{n}} - \frac{u_{f,i}\Delta t}{\Delta x}(\overline{T_{f,i}^{n}} - \overline{T_{f,i-1}^{n}}) + \frac{\alpha_f\Delta t}{\Delta x^2}(\overline{T_{f,i+1}^{n}} - 2\overline{T}_{f,i}^{n} + \overline{T_{f,i-1}^{n}}) \tag{2.5}$$

$$\overline{T_{s,i}^{n+1}} = \overline{T_{s,i}^{n}} + \frac{\alpha_s\Delta t}{\Delta x^2}(\overline{T_{s,i+1}^{n}} - 2\overline{T_{s,i}^{n}} + \overline{T_{f,s-1}^{n}}) \tag{2.6}$$

For the first fluid cell adjacent to the left boundary (by neglecting the conductive fluxes), the discrete equation can be written as:

$$\overline{T_{f,0}^{n+1}} = \overline{T_{f,0}^{n}} - \frac{u_{f,0}\Delta t}{\Delta x}(\overline{T_{f,0}^{n}} - \overline{T_{c}^{n}}) + \frac{\alpha_f \Delta t}{\Delta x^2}(\overline{T_{f,1}^{n}} - \overline{T}_{f,0}^{n}) \tag{2.7}$$

A similar equation for the case of discharging period can be derived, wherein a FTFS approach has to be used (negative $u_{f,i}$) according to the stability observed by modified wavenumber analysis. Further, for that case, the boundary condition will be imposed at the right boundary (T_d).

The Code verification plots for this case have been obtained in Section - *Code Verification Studies for Part 3*, described later.

In the second step, due to the stability restrictions on using explicit method, an implicit method may seem beneficial. Using a fully-implicit method necessitates solving a huge sparse system of equations, which is a factor of the number of cells, that will be computationally expensive. Hence, an alternative simplified version, namely point-implicit method which only makes use of an implicit source term in the equation has been adopted.
With this the discretized equations to be solved include :

$$\overline{T_{f,i}^{n+1}} = \overline{T_{f,i}^{n}} - \frac{u_{f,i}\Delta t}{\Delta x}(\overline{T_{f,i}^{n}} - \overline{T_{f,i-1}^{n}}) + \frac{\alpha_f \Delta t}{\Delta x^2}(\overline{T_{f,i+1}^{n}} - 2\overline{T}_{f,i}^{n} + \overline{T_{f,i-1}^{n}}) - h_{v,f}\Delta t(\overline{T_{f,i}^{n+1}} - \overline{T_{s,i}^{n+1}}) \tag{2.8}$$

$$\overline{T_{s,i}^{n+1}} = \overline{T_{s,i}^{n}} + \frac{\alpha_s \Delta t}{\Delta x^2}(\overline{T_{s,i+1}^{n}} - 2\overline{T_{s,i}^{n}} + \overline{T_{f,s-1}^{n}}) - h_{v,s}\Delta t(\overline{T_{s,i}^{n+1}} - \overline{T_{f,i}^{n+1}}) \tag{2.9}$$

Rearranging the equations gives us :

$$(1+h_{v,f}\Delta t)\overline{T_{f,i}^{n+1}} - h_{v,f}\Delta t\overline{T_{s,i}^{n+1}} = \overline{T_{f,i}^{n}} - \frac{u_{f,i}\Delta t}{\Delta x}(\overline{T}_{f,i}^{n} - \overline{T_{f,i-1}^{n}}) + \frac{\alpha_f \Delta t}{\Delta x^2}(\overline{T_{f,i+1}^{n}} - 2\overline{T_{f,i}^{n}} + \overline{T_{f,i-1}^{n}}) \tag{2.10}$$

$$(1 + h_{v,s}\Delta t)\overline{T_{s,i}^{n+1}} - h_{v,s}\Delta t\overline{T_{f,i}^{n+1}} = \overline{T_{s,i}^{n}} + \frac{\alpha_s \Delta t}{\Delta x^2}(\overline{T}_{s,i+1}^{n} - 2\overline{T_{s,i}^{n}} + \overline{T_{f,s-1}^{n}}) \tag{2.11}$$

These equations can be solved in two steps as detailed below:

1. Solving Equations 2.10, 2.11 by an explicit method (detailed earlier) to compute $\overline{T_{f,i}^{*}}$ and $\overline{T_{s,i}^{*}}$

2. These are then used to compute $\overline{T_{f,i}^{n+1}}$ and $\overline{T_{f,i}^{n+1}}$, the temperatures in the next time step by solving the 2 $\times$ 2 matrix at each cell.

$$\overline{T_{f,i}^{*}} = \overline{T_{f,i}^{n}} - \frac{u_{f,i}\Delta t}{\Delta x}(\overline{T_{f,i}^{n}} - \overline{T}_{f,i-1}^{n}) + \frac{\alpha_f \Delta t}{\Delta x^2}(\overline{T}_{f,i+1}^{n} - 2\overline{T_{f,i}^{n}} + \overline{T_{f,i-1}^{n}}) \tag{2.12}$$

$$\overline{T_{s,i}^{*}} = \overline{T_{s,i}^{n}} + \frac{\alpha_f \Delta t}{\Delta x^2}(\overline{T_{s,i+1}^{n}} - 2\overline{T}_{s,i}^{n} + \overline{T_{s,i-1}^{n}}) \tag{2.13}$$

2.2 Test Conditions

The developed numerical code was tested using the conditions representative of a thermal energy storage comprising of molten salt as the fluid and rocks as the solid phase. The parameters used for testing the routine and MMS are listed below :
$T_{f,0}$ *and* $T_{s,0} = 288\ K$
$T_c = 773\ K$
$\alpha_f = 2 \times 10^{-7}$
$\alpha_s = 9 \times 10^{-7}$
$\epsilon = 0.4$
$u_{f,i} = 0.1\ m/s$
$h_v = 1000\ W/m^3K$ (zero for Part 3)

2.3 Code Verification Plots and Discussions

2.3.1 Formulations

The actual order of accuracy can be computed from this formula as follows [1] :

$$Actual\ order\ of\ accuracy\ =\ \frac{log(\frac{E_3\ -\ E_2}{E_2\ -\ E_1})}{logr} \tag{2.14}$$

where E_3, E_2, E_1 are errors computed as the difference between the exact solution (manufactured) and the numerical solution. Further, r is the grid refinement ratio, which is taken to be the ratio of the successive grid spacings. It is important to note that Equation 2.14 is valid only in the asymptotic range of convergence, when the leading term in the error dominates the subsequent terms. The order of verification study was performed for two cases :

1. The discrete equations using the explicit Euler method (with h_v equal to zero) - Part 3

2. The discrete equations using point-implicit approach (coupled equations with non-zero h_v) - Part 4

Further, three types of error norms have been used in the order of verification plots : L_1 , L_2 and L_∞. The formulations for these types of error norms can be found in [1]. Further, the error used to calculate these error norms has been computed as :

$$Error\ =\ |T_i\ -\ T_{i,exact}| \tag{2.15}$$

For the current work, cosinusoidal functions which are a function of wavenumbers k, have been used for MMS.

$$T(x) \;=\; cos(kx) \tag{2.16}$$

where k is the wavenumber, which is a function of n, as :

$$k \;=\; \frac{2\pi n}{H} \tag{2.17}$$

Since the order of magnitude of this trigonometric function (used for manufactured solutions) is unity, the denominator of the error calculation in Equation 2.15 has been neglected [1]. Here, T_i and T_{exact} refer to the fluid or solid temperatures and exact temperatures calculated at location i.

2.3.2 Code Verification Studies for Part 3

As seen in Figure 2.1, the actual order of accuracy numerically obtained for the case of fluid is 0.99512 (L_1-error norm), 0.99713 (L_2-error norm) and 1.00614 (L_∞ error norm). For the case of fluid, in the spatially discretized terms, the diffusion term which is centrally differenced is second order accurate while the advection term which uses backward differencing scheme (upwind) is first order accurate. So, the overall nominal order of accuracy for the case of spatial discretization should be one [2]. Additionally, the time-integration method (Forward Euler) is first order accurate. Thus, the overall order of accuracy is first order considering the spatial and time-integration methods into account. Thus, the actual order of accuracy as seen in this Figure is consistent with the nominal order of accuracy for the fluid temperatures.
However, in the case of solid, the spatially discretized term is second order accurate as its centrally differenced, whereas the time-integration method is first order accurate. So, at large diffusion numbers (with large time time steps and large time-integration errors), the expected actual order of accuracy is 1, whereas at low diffusion numbers, it will be 2 (as the time-steps will be less, thus their time-integration errors will be small). As seen in Figure 2.2, the actual order of accuracy numerically obtained for the case of solid is 2.00187 (L_1-error norm), 2.00184 (L_2-error norm) and 2.00178 ($L - \infty$ error norm) at low diffusion numbers, which is consistent with the nominal order of accuracy.

2.3.2.1 Code Verification Studies for various Peclet numbers

Peclet number (Pe) is defined as the ratio of the convection and diffusion terms, and denotes the relative dominance between these two terms depending on the magnitude of Pe. At large values of Pe, the convection term dominates over the diffusion, whereas the converse is true for low Pe. But at Pe around unity, the order of magnitudes of convection and diffusion terms will become equal.Thus, for this study, three cases of Pe were considered : 100, 1 and 10^{-3}, to signify the cases when the relative dominance between the diffusion and convection term holds. Additionally, three different values of n (1,3 and 5) were used in the calculation of wavenumbers for manufactured solutions.

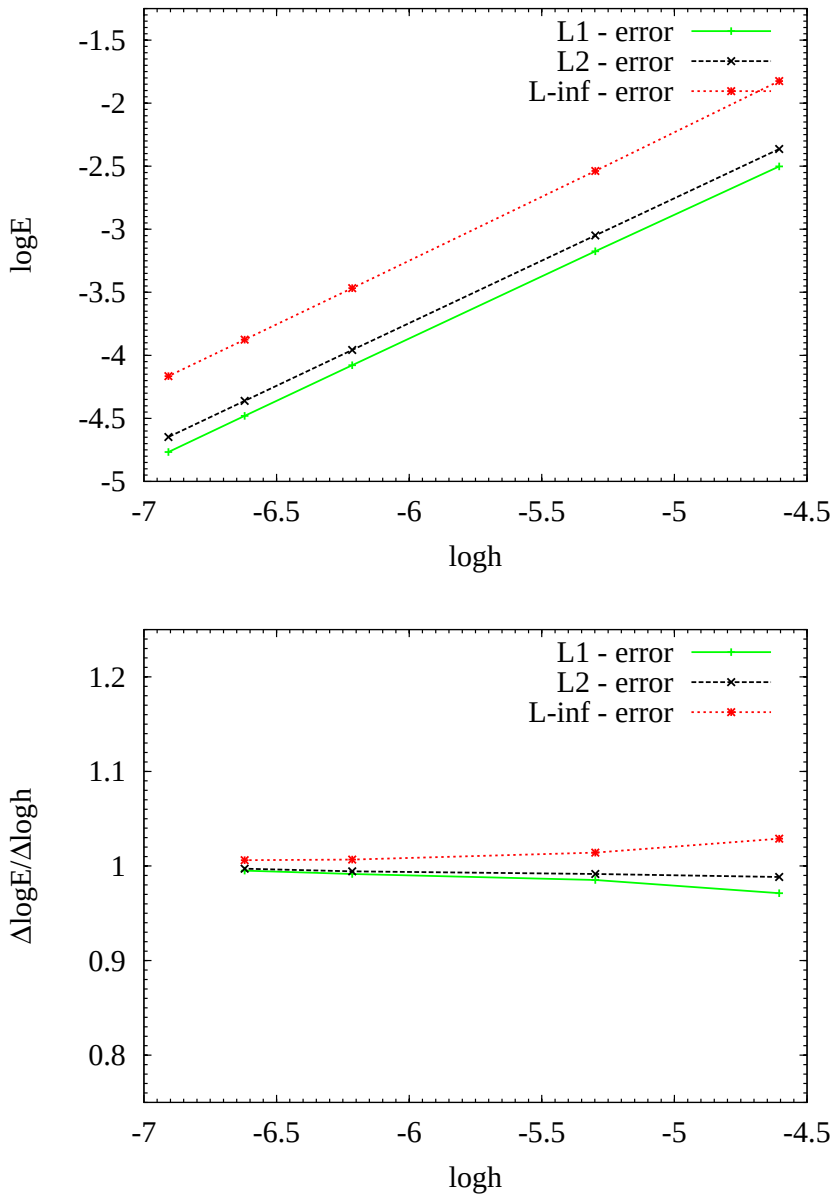

Figure 2.1: Order of verification studies using the Method of Manufactured solutions for fluid temperatures (Part 3)

By neglecting the source and transient terms, the equations for the mathematical model of the fluid can be written in terms of Pe as:

$$\frac{\partial T_f}{\partial x} = Pe^{-1}\frac{\partial^2 T_f}{\partial x^2} \tag{2.18}$$

where

$$Pe = \frac{u_{f,i}H}{\alpha_f} \tag{2.19}$$

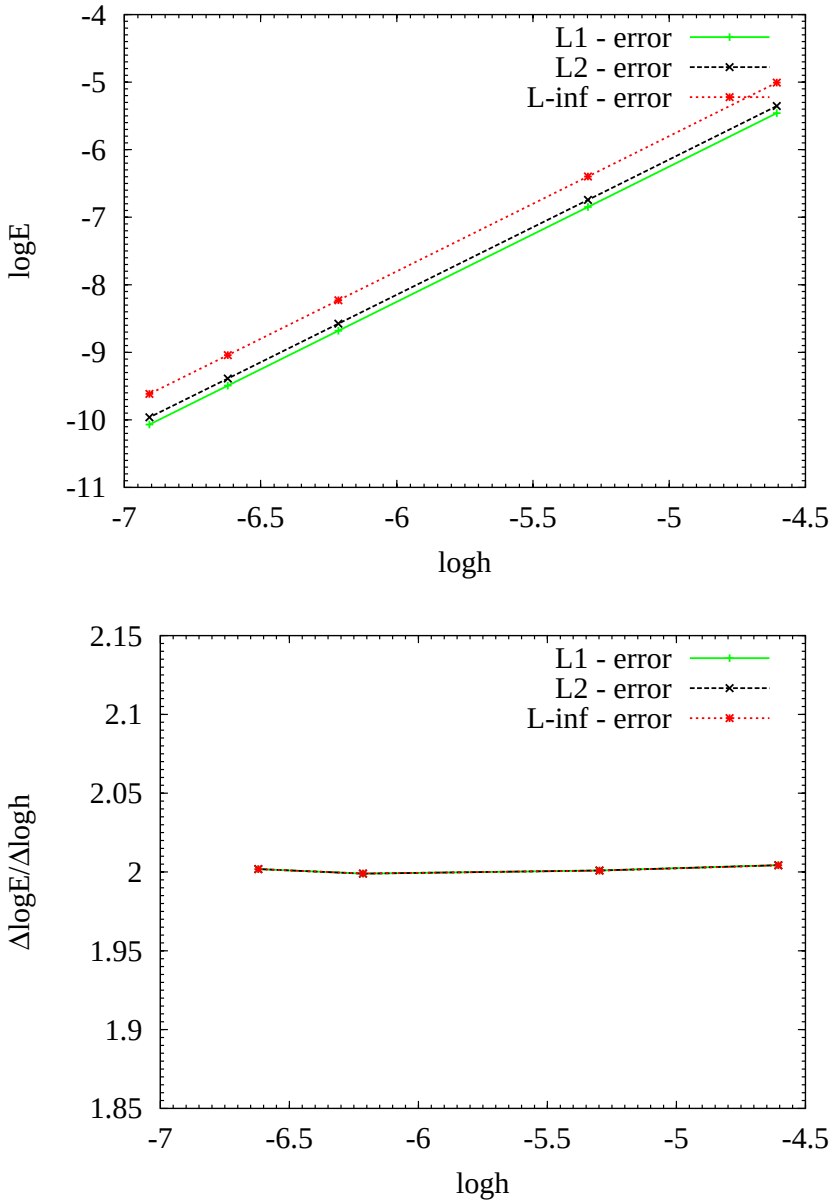

Figure 2.2: Order of verification studies using the Method of Manufactured solutions for solid temperatures at low diffusion number (Part 3)

Peclet number = 10^{-3}

At low values of Pe (in this case 10^{-3}), the diffusion term will become more significant (as Pe^{-1} will become large) than the convection term. Hence, the overall order of accuracy will be 2 (for the case of low diffusion numbers, by neglecting the time-integration errors). The actual order of accuracy obtained for this case with various values of n, as seen in Figures 2.3 - 2.5 is thus consistent with the theoretical values.

Peclet number = 100

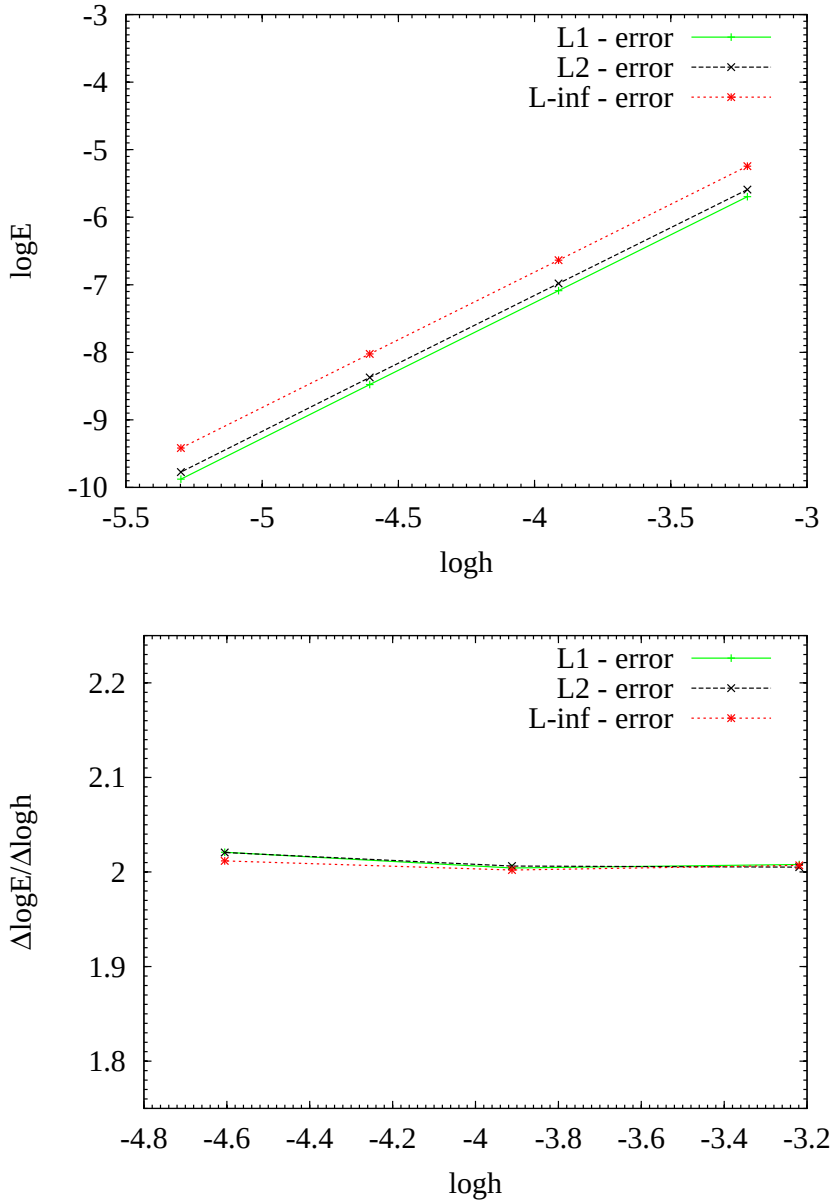

Figure 2.3: Order of verification studies using MMS for fluid temperatures at Pe = 10^{-3}, $n = 1$

At high values of Pe (in this case 100), the convection term will become more significant (as Pe^{-1} will become small) than the diffusion term. Hence, the overall order of accuracy will be 1. The actual order of accuracy obtained for this case with various values of n, as seen in Figures 2.6 - 2.8 is thus consistent with the theoretical values.

Peclet number = 1

This case is an interesting situation, as both the terms - convection and diffusion will become significant. In such a case, both these terms will offer opposing

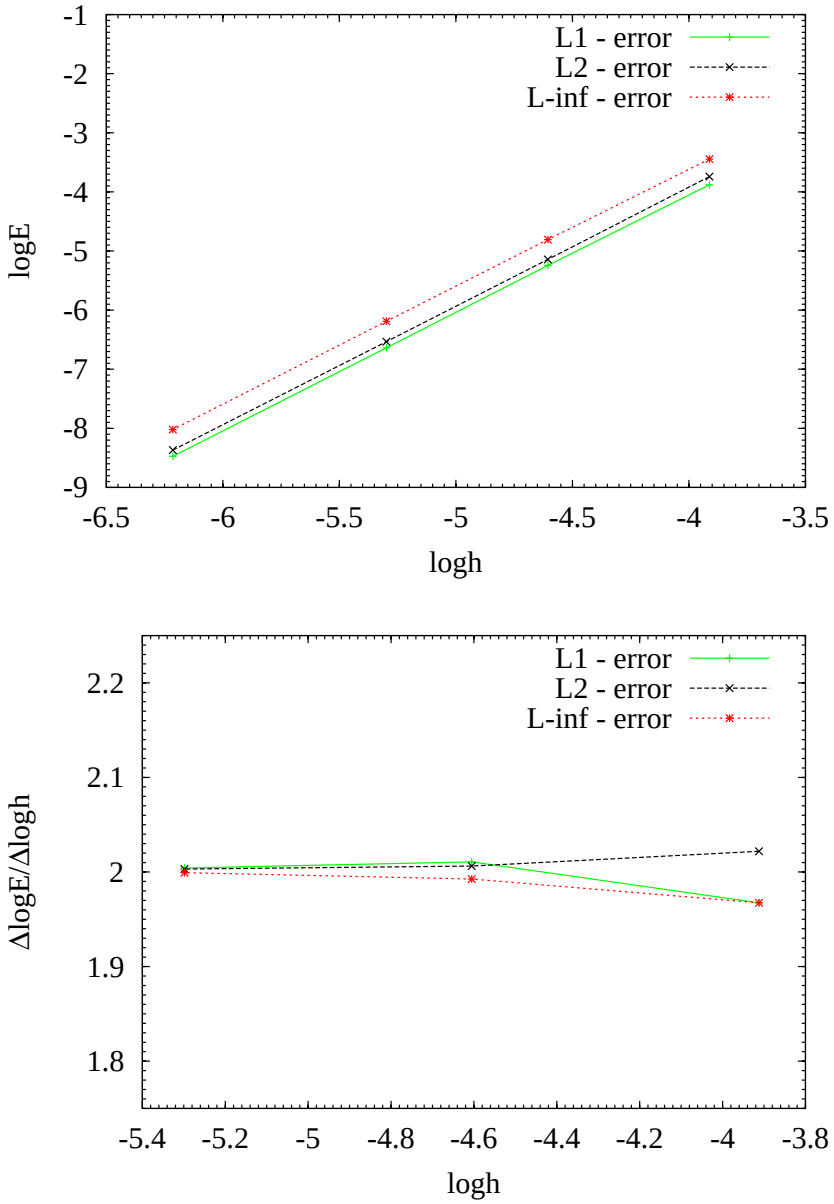

Figure 2.4: Order of verification studies using MMS for fluid temperatures at Pe $= 10^{-3}$, $n = 5$

effects, thus resulting in orders of accuracy close to 2 at high grid spacings and close to 1 at low grid spacings. It can be seen in the Figure 2.9 that the actual order of accuracy is in the range of 1 to 2, which is consistent with the theoretical observations.

Since the present code with zero h_v has been successfully verified using MMS, it can be now extended to handle non-zero h_v.

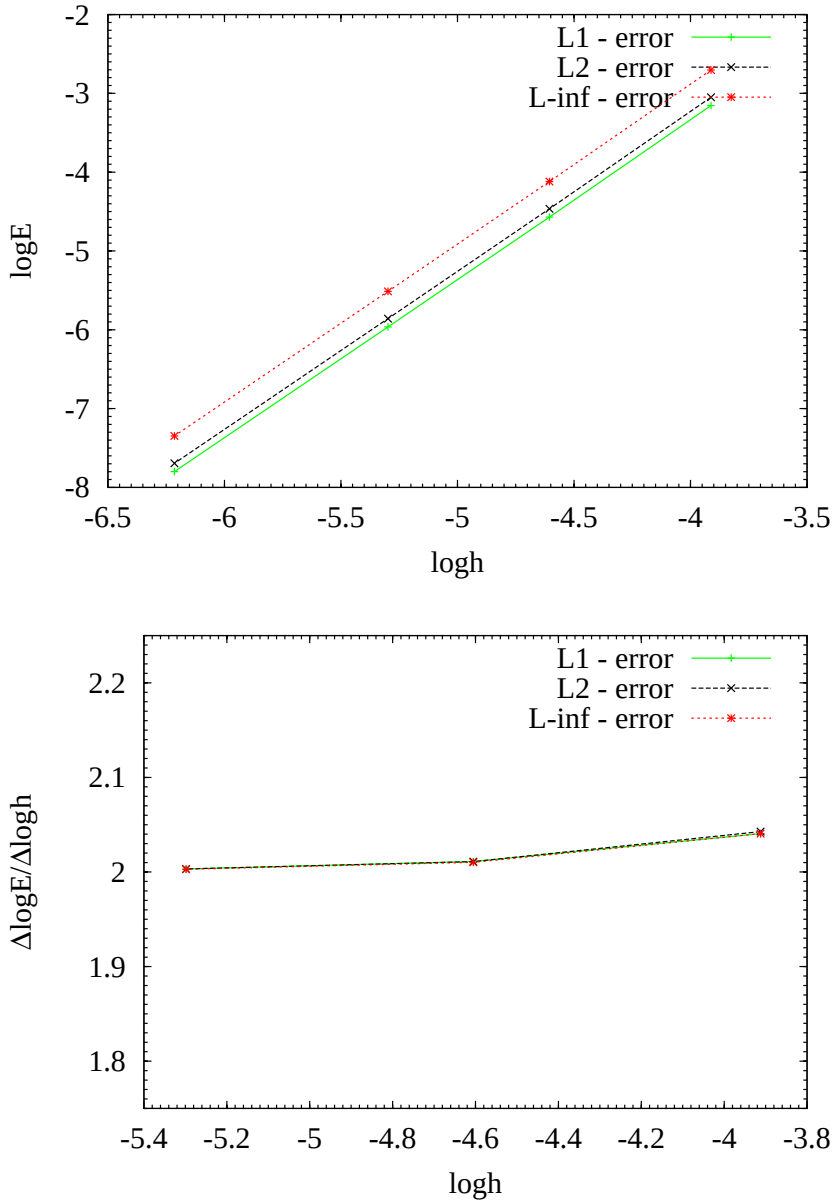

Figure 2.5: Order of verification studies using MMS for fluid temperatures at Pe $= 10^{-3}$, $n = 7$

2.3.3 Code Verification Studies for Part 4

On a similar analogy as discussed earlier, the nominal order of accuracy for the fluid temperatures must be 1. Since both the temperature equations (solid and fluid) are coupled, the nominal order of accuracy for the case of solid will also be 1. As seen in Figure 2.10, the actual order of accuracy numerically obtained for the case of fluid is 1.00098 (L_1-error norm), 1.00092 (L_2-error norm) and 1.00402 (L_∞ error norm). Additionally, the actual order of accuracy numerically obtained for the case of solid is 1.00126 (L_1-error norm), 1.00163

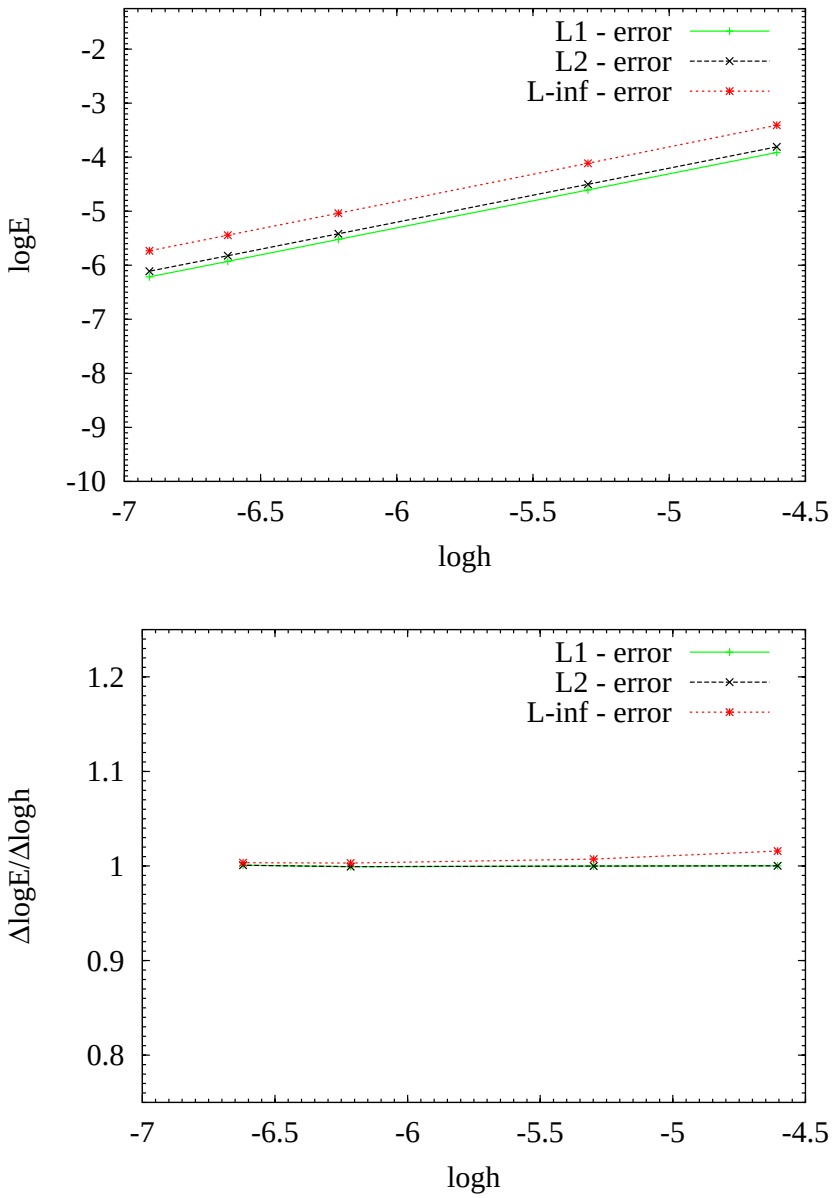

Figure 2.6: Order of verification studies using MMS for fluid temperatures at Pe = 100, $n = 1$

(L_2-error norm) and 1.02371 (L_∞ error norm) as seen in Figure 2.11. For both the cases, the considered grid spacings render the plots to be in the asymptotic range of convergence, causing the actual order of accuracy to approach 1 as the grid spacing is decreased. Thus, the results obtained are consistent with the theoretical ones.

Its important to note that for this case two manufactured solutions (separate ones for solid and fluid phases) have been considered, with wavnumbers k and 2k. This is to ensure that the source terms do not get canceled out during the

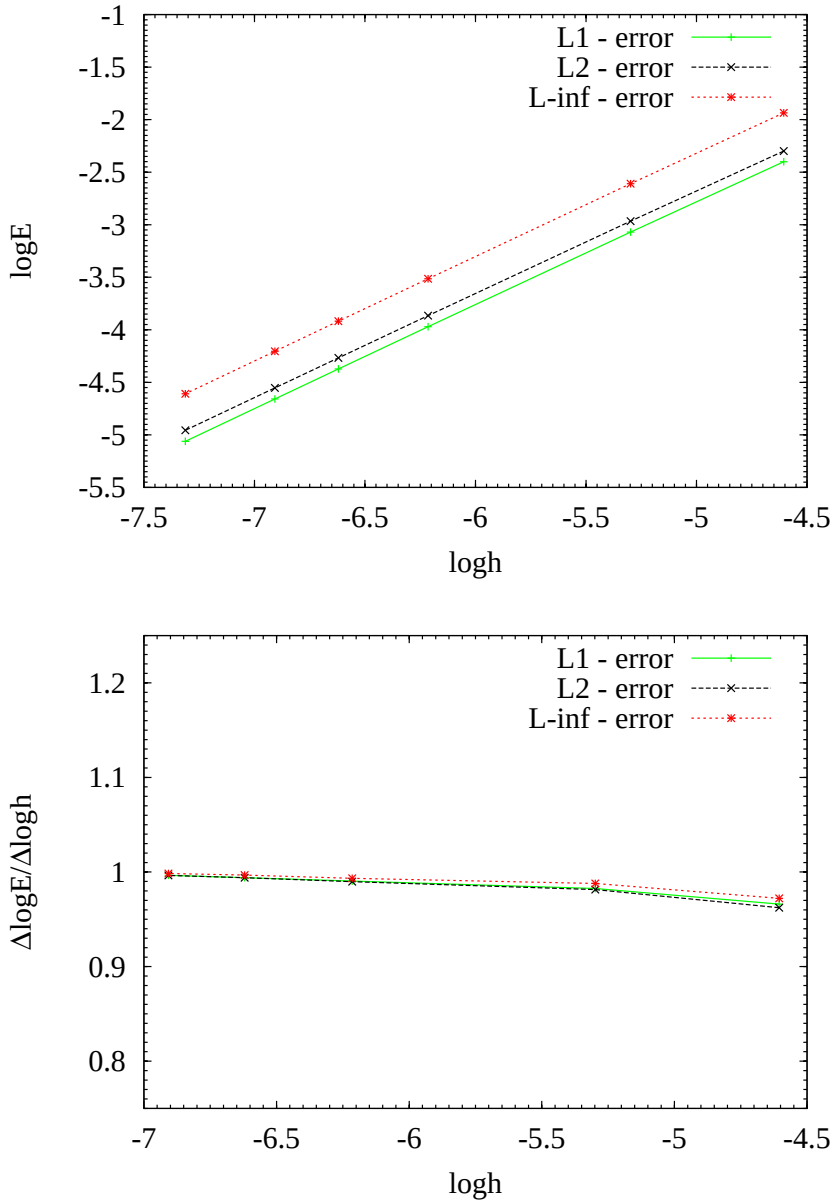

Figure 2.7: Order of verification studies using MMS for fluid temperatures at Pe = 100, $n = 5$

calculations, as both the solid and fluid temperature equations are coupled.

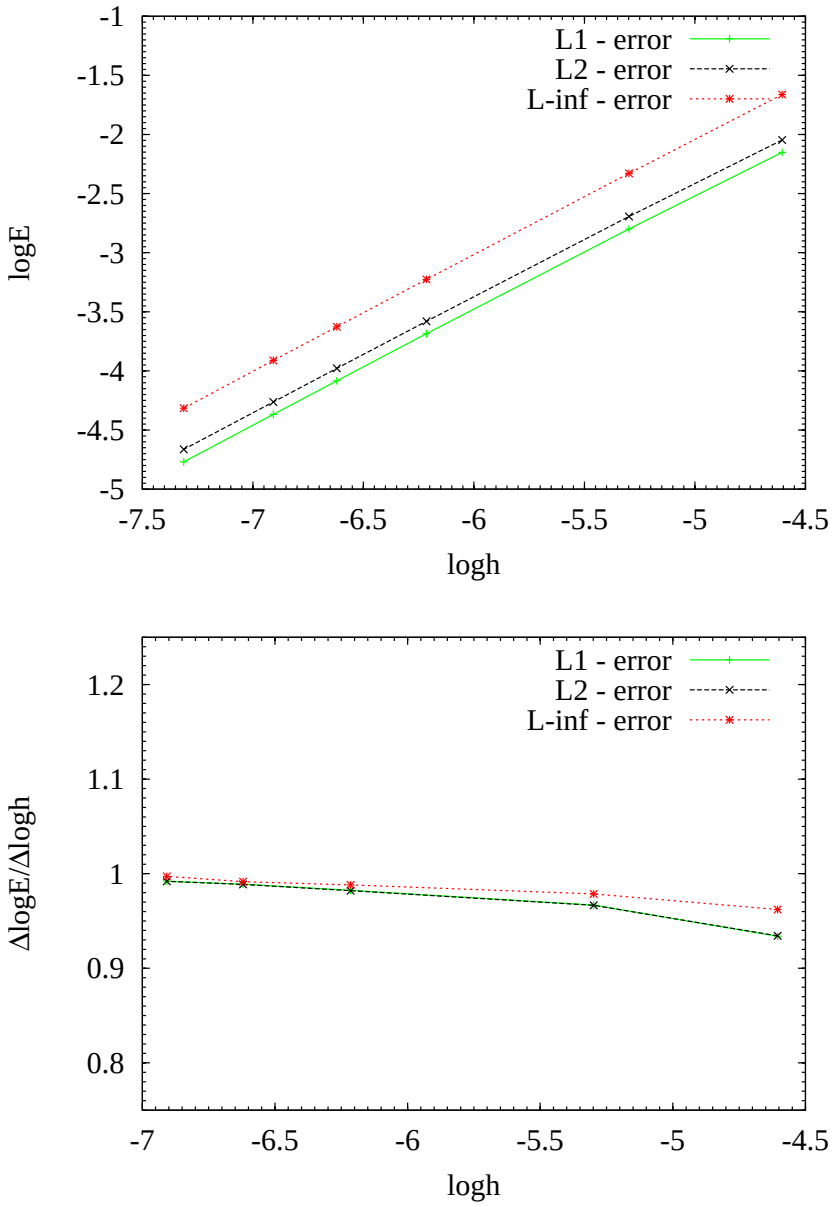

Figure 2.8: Order of verification studies using MMS for fluid temperatures at Pe = 100, $n = 7$

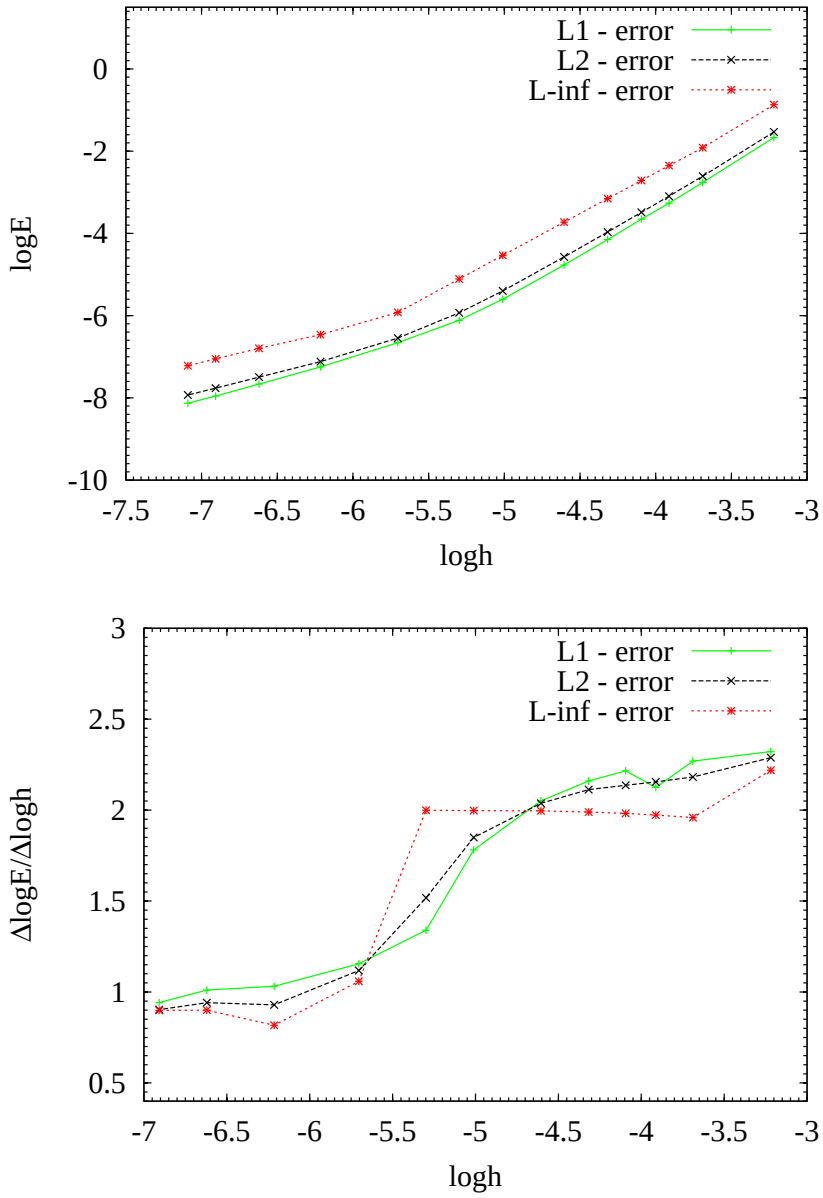

Figure 2.9: Order of verification studies using MMS for fluid temperatures at Pe = 1, $n = 7$

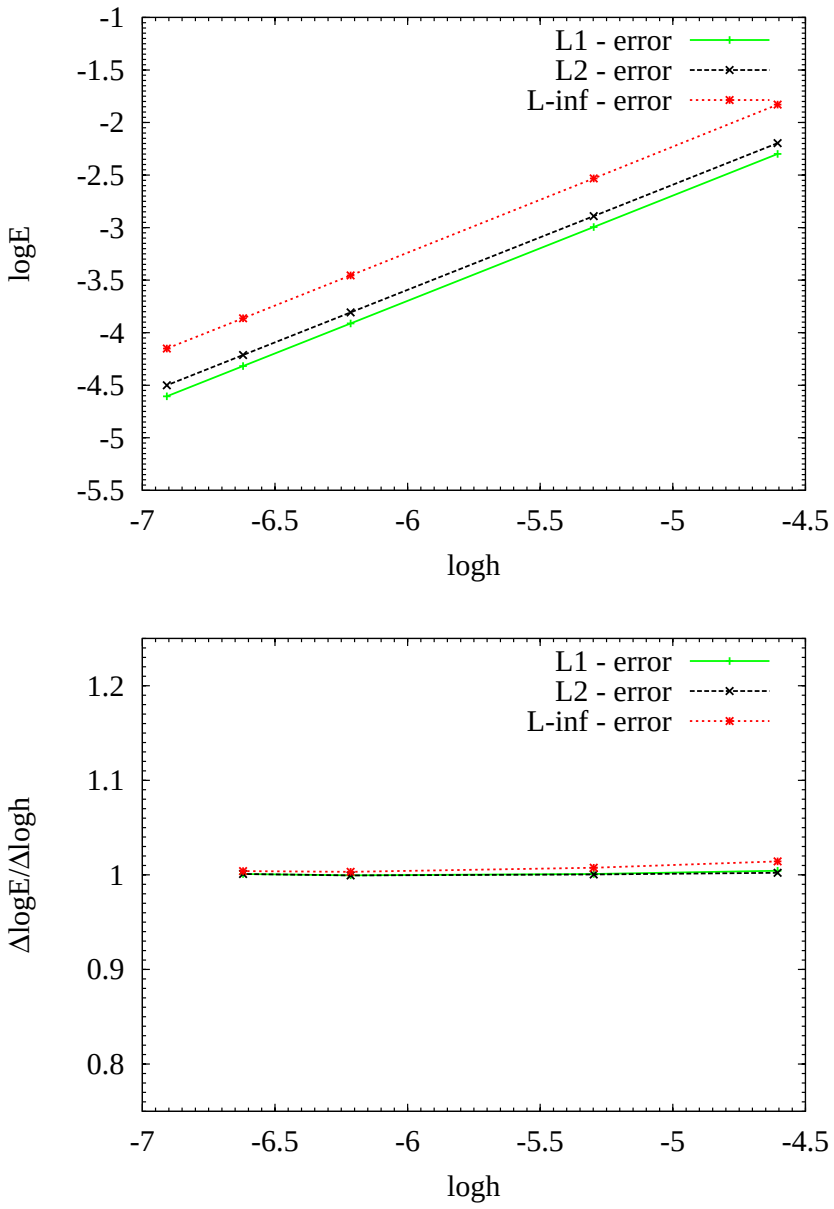

Figure 2.10: Order of verification studies using the Method of Manufactured solutions for fluid temperatures (Part 4)

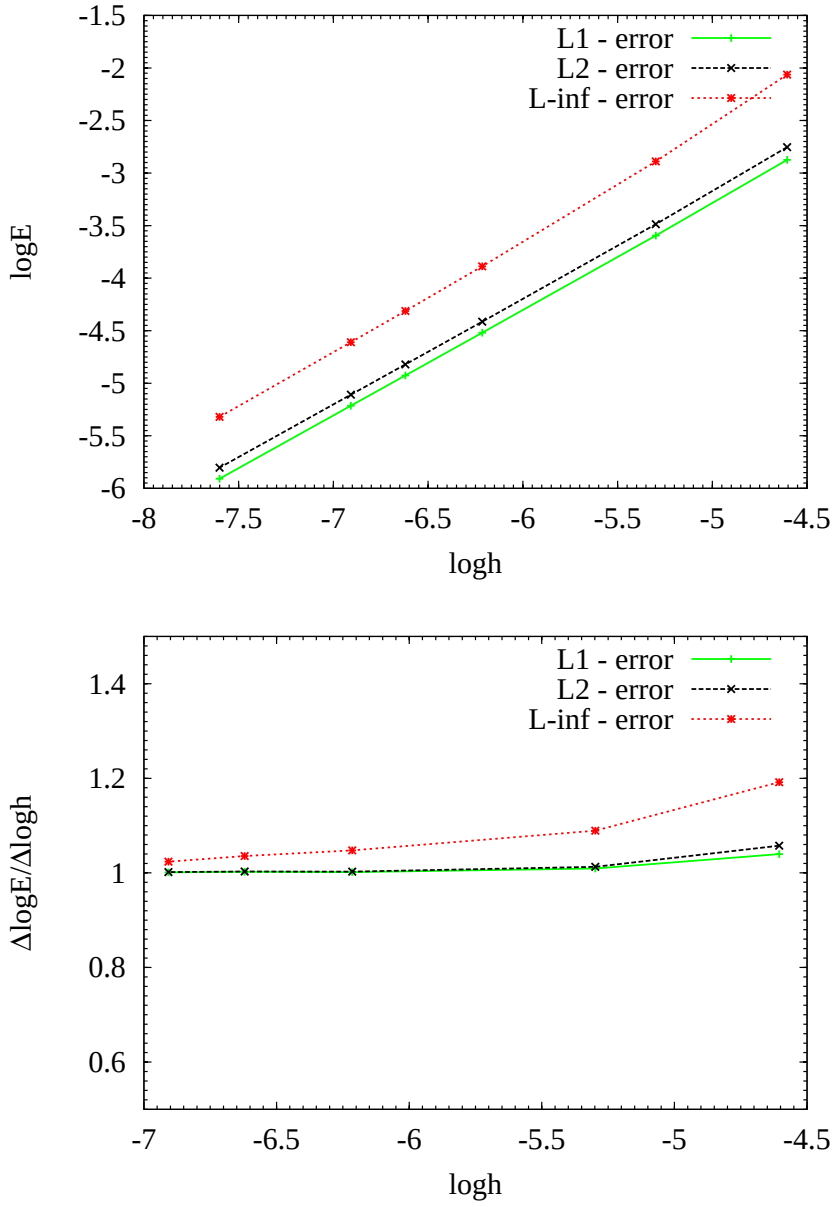

Figure 2.11: Order of verification studies using the Method of Manufactured solutions for solid temperatures (Part 4)

Chapter 3

Code Verification Studies using the Exact solution

3.1 Exact solution of the simplified equations

The numerical solutions obtained by solving the fully discrete equations as mentioned before were compared with the exact solution (obtained analytically) for code verification.
The conduction terms have been neglected in this case, and non-dimensional coordinates have been used as follows :

$$\zeta(x) \; = \; \frac{h_v x}{\epsilon \rho_f C_{p,f} u_{f,i}} \tag{3.1}$$

$$\eta(x,t) \; = \; \frac{h_v}{(1 \; - \; \epsilon)\rho_s C_s}(t \; - \; \frac{x}{u_{f,i}}) \tag{3.2}$$

The non-dimensional temperature can be defined as :

$$\overline{T} \; = \; \frac{T \; - \; T_{s,0}}{T_c \; - \; T_{s,0}} \tag{3.3}$$

The governing equations can be simplified to :

$$\frac{\partial \overline{T_f}}{\partial \zeta} \; = \; \overline{T_s} \; - \; \overline{T_f} \tag{3.4}$$

$$\frac{\partial \overline{T_s}}{\partial \eta} \; = \; \overline{T_f} \; - \; \overline{T_s} \tag{3.5}$$

The exact solution of these equations has been shown as :

$$\overline{T_f}(\zeta,\eta) \; = \; 1 \; - \; e^{-\eta} \int_0^{\zeta} e^{-\zeta'} J_0(2i\sqrt{\zeta'\eta}) d\zeta' \tag{3.6}$$

$$\overline{T_s}(\zeta,\eta) \; = e^{-\eta} \int_0^{\eta} e^{-\eta'} J_0(2i\sqrt{\zeta\eta'}) d\eta' \tag{3.7}$$

where J_0 is the zeroth order Bessel function.

3.2 Parameters and formulations for the Numerical solution

To compare the results of the numerical solution (approximate) with the exact solution, the following physical parameters were chosen (code verification was performed only for the charging phase at 5000 sec).
$\epsilon = 0.4$
$d_s = 0.03\ m$
$\rho_s = 2600\ kg/m^3$
$\rho_f = 1835.6\ kg/m^3$
$C_s = 900.0\ J/kgK$
$C_{p,f} = 1511.8\ J/kgK$
$k_s = 2.0\ W/mK$
$k_f = 0.52\ W/mK$
$\mu_f = 2.63\ kg/ms$
$m_f = 0.1\ kg/s$
$T_{s,0} = 293\ K$
$T_c = 873\ K$

The volumetric heat transfer coefficient can be computed from

$$h_v = \frac{6(1 - \epsilon)}{d_s} h \tag{3.8}$$

The overall heat transfer coefficient in Equation 3.8 can be shown as :

$$h = (\frac{1}{h_{f,s}} + \frac{d_s}{10k_s})^{-1} \tag{3.9}$$

The fluid-solid heat transfer coefficient in Equation 3.9 can now be calculated as :

$$h_{f,s} = Nu_{f,s}\frac{k_f}{d_s} \tag{3.10}$$

Its corresponding Nusselt number can be given as :

$$Nu_{f,s} = \frac{0.255}{\epsilon} Pr^{1/3} Re^{2/3} \tag{3.11}$$

The Prandtl and Reynolds numbers to be used in the above expression may be given as:

$$Pr = \frac{\mu_f C_{p,f}}{k_f} \tag{3.12}$$

$$Re = \frac{\epsilon \rho_f u_{f,i} d_s}{\mu_f} \tag{3.13}$$

3.3 Code verification plots and Discussion

The code verification plot was obtained for several grid spacings by plotting the temperature profiles, as seen in Fig 3.1. It can be thus seen that the profile obtained by the numerical solution is quite in good agreement to the one by the exact solution, though small deviation is observed near the transition where the profile of the fluid temperature develops (thermal front). This is because of the fact that the conduction terms have been neglected in deriving the simplified exact solution, though they have been considered in the discrete equations. Even if one happens to neglect the conduction terms in these discrete equations, small deviation can be still noticed at this thermal front due to the usage of an upwinding differencing scheme (Backward) for the advection equation according to the principle of modified wavenumber analysis (the real part of the modified wavenumber will be positive for backward difference of the first derivative - which is responsible for causing the effect of numerical diffusion or dissipation).

Good agreement of the results obtained by the numerical solution thus verifies the credibility of the developed numerical code, which can be later used for performing large scale simulations representing the exact thermal energy storage (as done in Chapter 4).

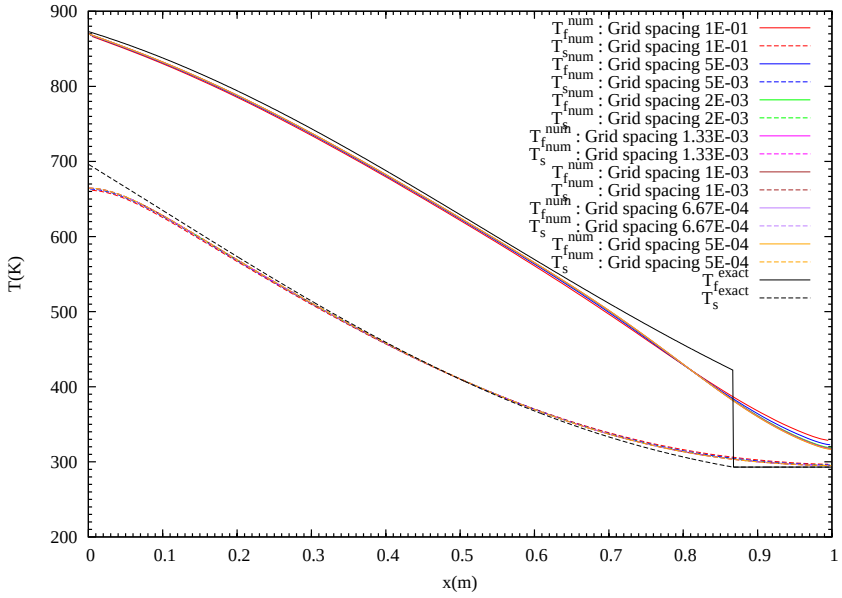

Figure 3.1: Code verification with the exact solution

Chapter 4

Simulation results of the Storage Design

4.1 Numerical Simulations

4.1.1 Paramters and Formulations

The following parameters have been chosen for the current numerical study:

$V = 300\ m^3$
$t_c = 6$ hours
$t_d = 6$ hours
$t_i = 6$ hours
$\epsilon = 0.4$
$d_s = 0.03$ m
$\rho_s = 2600\ kg/m^3$
$\rho_f = 1835.6\ kg/m^3$
$C_s = 900\ J/kgK$
$C_{p,f} = 1511.8\ J/kgK$
$k_s = 2.0\ W/mK$
$k_f = 0.52\ W/mK$
$\mu_f = 2.63\ kg/ms$
$m_f = 10\ kg/s$
$T_c = 873$ K
$T_d = 293$ K

Additionally, five cases of storage diameters were used : $4 - 8$ m in increments of 1 m. To ensure that the storage volume remains constant, the height in the simulation was input accordingly in `void_user_input()` function in the code. To assess the performance of the storage, the cycle exergy efficiency has been computed using the formula :

$$\eta = \frac{E_{d,out} - E_{d,in}}{E_{c,out} - E_{c,in}} \tag{4.1}$$

where E, out, in, c and d refer to the exergy flux, inflow condition, outflow condition, charging and discharging periods respectively.

The exergy flux can now be calculated as :

$$E = \int mC_{p,f}[T\ - T_0\ - T_0 \ln\left(\frac{T}{T_0}\right)] \tag{4.2}$$

where T_0 is taken to be the reference temperature of 288.15 K.
Additionally, the capacity factor was calculated as :

$$Capacity\ Factor\ =\ \frac{Actual\ Thermal\ Energy\ Stored\ (Q(t))}{Maximum\ Thermal\ Energy\ Stored\ (Q_{max})} \tag{4.3}$$

The actual thermal energy stored $Q(t)$ can be expressed as :

$$Q(t)\ =\ \frac{\pi}{4}D^2[\epsilon\rho_f C_{p,f} \int (T_f(z,t)\ -\ T_d)dz\ + (1\ -\ \epsilon)\rho_s C_s \int (T_s(z,t)\ -\ T_d)dz] \tag{4.4}$$

Additionally, the maximum thermal energy can be computed as :

$$Q_{max}\ =\ [\epsilon\rho_f C_{p,f}\ +\ (1\ -\ \epsilon)\rho_s C_s]\frac{\pi}{4}D^2 H(T_c\ -\ T_d) \tag{4.5}$$

The cycles were incremented as long as the difference between the exergy efficiencies of two consecutive cycles turned out to be less than $1E-05$. After reaching this criterion, the capacity factor, cycle exergy efficiency and temperature increase at the outflow were computed. Note that the temperature increase at the outflow was taken to be the difference between T_d and the temperature computed at outflow.

4.1.2 Grid Refinement Studies

To ensure the independence of simulation results on the changes in grid spacing, temperature increase at the outflow was plotted for several grid spacings. Then, the optimum grid spacing below which the change in the temperature increase would be insignificant was determined. This study was performed for all the storage diameters as seen in Figure 4.1. As seen in this Figure, an optimum grid spacing of 0.005 is enough for the storage diameters of 6, 7 and 8 m. However, a larger grid spacing of 0.01 can be chosen for the case of storage diameters equal to 4 and 5 m (as the rate of decrease of the temperature difference with the grid spacing is less than that of the other diameters) .

4.2 Results of various storage diameters

Using the grid spacings determined earlier, the simulations were performed and the necessary parameters - cycle exergy efficiency, temperature increase at outflow and capacity factor were computed. These are mentioned in the Table 4.1.

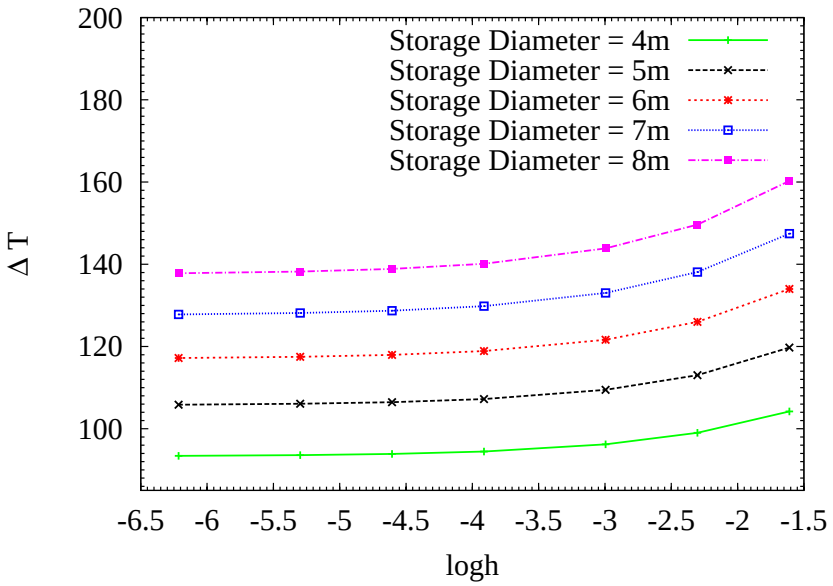

Figure 4.1: Grid refinement study to determine the optimum grid spacing to be used for each storage diameter

Additionally, the fluid temperature profile during the first cycle for the storage diameter of 8 m was plotted in Figure 4.2. It can be seen that the thermal front of the fluid develops and progresses in the positive direction with the fluid interstitial velocity $u_{f,i}$ at times of $1E03$ and $2E03$ seconds. However, the thermal front does not progress during the idle phase at times of $3E03$ and $4E03$ seconds, as interstitial fluid velocity remains zero. However, due to diffusion, small movements in the thermal front will be plausible (not clearly visible in this Figure, as both the profiles coincide). Contrast to the charging period, the thermal front in the discharging period progresses in the negative x-direction, as seen at times of $5E03$ and $6E03$ seconds

Table 4.1: Results of the storage design for all storage diameters

Diameter(m)	**Temperature increase(K)**	**Exergy Efficiency**	**Capacity Factor**
4.0	93.410527	0.953611	0.416750
5.0	105.843037	0.941253	0.412035
6.0	117.213379	0.928928	0.407193
7.0	127.808298	0.916313	0.402148
8.0	137.828299	0.904062	0.397021

4.3 Possible improvements in the physical model

Since the physical model adopted in the current work makes use of many simplified assumptions, the following improvements in the physical model can be suggested in a possible future work :

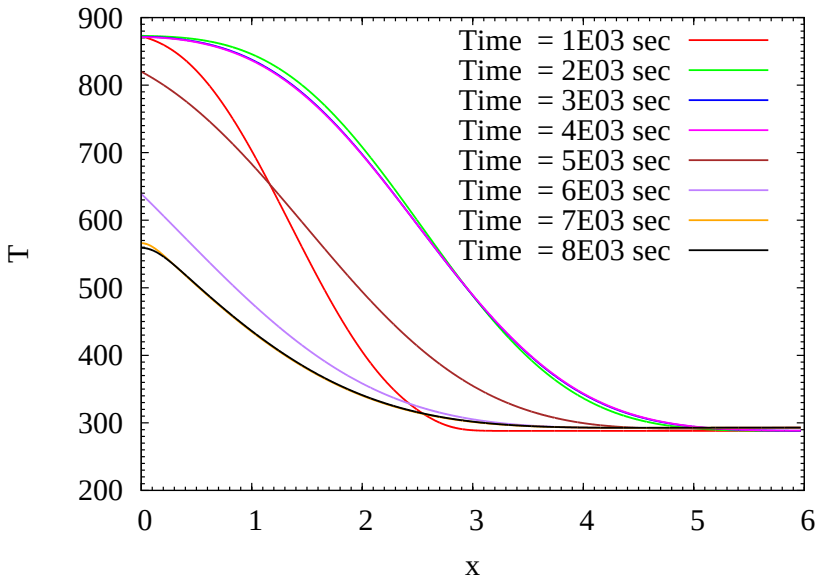

Figure 4.2: Variation of the fluid temperature during the first cycle for the storage diameter of 8m

1. Accounting the variation of these parameters (for fluid and solid phases) with temperature :

   - Density (ρ)
   - Thermal conductivity (k)
   - Specific heat at constant pressure (C_p) [3]
   - Volumetric heat transfer coefficient (h_v)

   These variations can be accounted in the numerical code by using polynomial fits or appropriate correlations from the literature. Apart from the temperature, the volumetric heat transfer coefficient may depend upon the mass flow of fluid and effective thermal conductivity (which can be a function of the radiative heat transfer coefficient to account for the radiation effects) also [3].

2. Considering the fluid to be compressible (non-constant density).

3. Considering the axial variation of porosity (or void fraction) of the solid rocks along the height of the storage [4]. This can be experimentally determined to account for the random positioning of the solid rocks (along with their shape and size).

4. It would be practically realistic to use a two-dimensional or three-dimensional physical model, rather than a simplified one-dimensional model adopted in the current work. For the former case, one may have to rely on commercial softwares as numerically modeling them will become intricately complex.

5. Accounting the convective heat losses through the walls [5], which is a function of the temperature difference between the fluid and ambient, and the overall wall heat transfer coefficient. It becomes cumbersome to calculate the heat transfer coefficient of the solid and fluid phases separately, to calculate the overall heat transfer coefficient. Rather, existing correlations could be taken from the literature.

6. Accounting the variations in the interstitial fluid velocity, as its magnitude cannot be constant throughout the cycle due to small fluctuations in the mass flow rate. Additionally, variable mass flow may be considered.

7. Considering the radiative heat transfer to the surroundings into account.

8. In the current work, a simplified assumption of constant fluid inlet temperature (during charging period) and constant fluid outlet temperature at the outlet (during discharging period) have been assumed, which need not be practically true. Accounting the changes in these temperatures is necessary for an accurate physical model.

4.4 Possible improvements in the Numerical Approximations

The following changes can be incorporated in the numerical model used in the current work :

1. Increasing the order of accuracy (spatial and time-integrations)

2. Using higher order accurate discretization methods for the conductive fluxes across the boundaries, rather than neglecting them in the current work.

4.4.1 Spatial Discretization

1. It would be better to use a central differencing scheme (second-order accurate) for discretizing the diffusion term (in the solid and fluid phase equations).

2. For the advective term (upwinding), a semi-Lagrangian method with unconditional stability might be preferable [6]. If one is using an implicit approach (time-integration) for the upwinding advective term, it may also be viable to use a deferred correction approach, which incorporates a first order upwind and a higher order approximation (such as QUICK scheme) [7, 8] to obtain higher accuracy.

4.4.2 Time-Integration

1. A trapezoidal (or Crank-Nicholson) method is preferable for the diffusion terms, as its second order accurate and unconditionally stable.

2. For the advection equation, Adams Bashforth 2 or Explicit Runge Kutta 2 can be used for attaining second order accuracy.

Additionally, to ensure second order accuracy across the boundaries, one may also use second-order accurate one-sided differencing scheme to discretize the conductive flux.

Its important to note that if one uses implicit (fully or trapezoidal) time-stepping schemes, it may become cumbersome to solve the sparse system of equations involving the fluid and solid temperatures. In that case, efficient methods to solve these system of equations is necessary.

If one wants to extend the current work to higher dimensions (two-dimensional or three-dimensional), fractional step (or operator splitting - time splitting in this context) methods may be useful. A possible alternative [1] to include this (Some inferences have been made from the literature pertaining to operator splitting methods in Advection-Diffusion-Reaction equations [9, 10, 11]) :

1. Numerically solving the advection equation separately
2. Numerically solving the diffusion equation concerning the fluid temperatures
3. Numerically solving the diffusion equation concerning the solid temperatures
4. Incorporating implicit source terms and updating the temperatures at the new time step (and solving the sparse system of equations simultaneously).

However, simple splitting methods may seem to be an inappropriate choice as they induce splitting errors of first order. In this view, higher order accurate splitting methods like Strang's second order accurate splitting method may seem useful. Solving the linear source terms (of solid and fluid) concerning temperatures in the last step of the aforementioned scheme may however not induce any splitting error, as they are linear.

[1] The author has made an attempt in reading relevant literature pertaining to operator splitting methods, but cannot assure the proposed scheme to work successfully, as it was beyond the scope of the current course

Chapter 5

General Description about the Code Structure and Working

All simulations were performed on one compute node of the Euler cluster [1], whose specifications are listed in [12]. Additionally, git version control system was used to keep a track of the necessary changes made in the code.

5.1 Structure of the code

The entire code comprises of the following:

1. Header file for function declarations (`function_list.h`) along with the global variables to be used in the program.

2. A file (`function_list.c`) which comprises of the function definitions of the respective functions defined in th header file. The file has been made as user-friendly as possible with necessary comments along with a brief description of each function (through `@brief` and `@params`).

3. A main function (`Thermal_Energy_Storage.c`) which calls the functions defined in `function_list.c` for performing the relevant simulations.

5.1.1 Working of each function

5.1.1.1 `calc_param_sim()`

The main objective of this function is to define extra parameters, specifically for the validation (Part 5) of the numerically obtained results with the exact solution. This is called by `user_input()` function, and the former can be commented out, while running the simulations for other parts of the Project (specifically MMS studies for Part 3 and 4).

[1] The code was not parallelized. The reason behind running the code in Euler was that I found it easier to submit the jobs directly to the cluster (which also informs me when the job has been done through an email), when the simulations consumed a few hours.

5.1.1.2 `user_input()`

This function either allows the simulation to proceed with the default values specified when the function parameter turns out to be 0, or allows the user to input values of his choice (for any non-zero function parameter). Alternatively, the user can edit the values when the function parameter is 0, and proceed with the simulation.

5.1.1.3 `gauss_elimination()`

In the part 4 of the project, point-implicit method is used. So, to solve the fluid and solid temperatures iteratively which arise due to the coupling, one has to solve a 2×2 matrix at each point to compute the fluid and solid temperature at new time steps. The two equations have been thus solved using Gaussian elimination [13]

5.1.1.4 `visualization()`

This function prints the solid and fluid temperatures at a particular iteration step, into a data file. To prevent the temperatures overwriting into the same file, the file name is indexed 5 integers long, in the end. This can later be used for obtaining the variations in the fluid and solid temperatures across the domain using Gnuplot (used in the current work).

5.1.1.5 `initialize()`

This function serves the purpose of initializing the fluid and solid temperatures initially. Separate wave-numbers are used for solid and fluid phases (`k_waveno1` for fluid and `k_waveno2` for solid). Different wavenumbers are essential for OVS in Part 4.

5.1.1.6 `status_storage()`

Based on the current time passed as the parameter to this function, an integer is returned depending on the condition if the storage medium is in charging, discharging or idle phase.

5.1.1.7 `liq_solve()`

This function computes the fluid temperatures at the next time step, in accordance with the discretized equations. `if` conditions are used to distinguish the equations from MMS to the normal simulation.

5.1.1.8 `solid_solve()`

This function is analogous to the `liq_solve()` function, except that it handles solid temperatures instead.

5.1.1.9 `coupled_solve_implicit()`

This function serves the purpose of incorporating the point-implicit method to solve the equations. For this purpose, it calls the `liq_solve()` and `solid_solve()` appropriately. Additionally, during the MMS studies, a separate source term factor is accounted for in this function also.

5.1.1.10 `error_compute_ovs_solid()`

This function computes the L_1, L_2 and L_∞ error norms for the solid temperatures, necessary for OVS.

5.1.1.11 `error_compute_ovs_liquid()`

This function serves the same purpose as `error_compute_ovs_solid`, except that it handles the fluid temperatures.

5.1.1.12 `conv_error()`

This function computes the error, to track the convergence (according to the convergence criterion specified by the user) necessary for OVS studies.

5.1.1.13 `q_max()`

This function computes the maximum thermal energy that can be stored, which uses Equation 4.5 for computation.

5.1.1.14 `q_time()`

This function uses Equation 4.4 to compute the thermal energy in the storage at any time instant, making it easier to calculate capacity factor.

5.1.1.15 `exergy_flux()`

This function uses Equation 4.2 to calculate the exergy flux, at any time instant. This can be easily used to compute the exergy flux at discharging, charging periods and inflow or outflow points of the computational domain.

5.1.2 Additional information about certain variables in the Code

To perform OVS, initialize `ovs_liquid` or `ovs_solid` to 1 (integer) depending on the order of fluid or solid discretized equations you want to compute. If you want to compute the order for both (as in the case when point-implicit approach was used with non-zero h_v - Part 4 of the Assignment), then both these variables may be initialized to 1. For OVS, one may uncomment the line pertaining to the calculation of `err` to obtain convergence on using MMS. However, for the full scale simulation, comment this convergence error criterion and uncomment the criterion of `ex_error` which is necessary to compute the cycle exergy efficiency (as the cycles have to be repeated as long as the difference between the successive exergy efficiencies are less).

5.2 Instructions for Compilation and Execution

GCC 4.8.2 compiler (available in Euler) was used for the current work. The compilation is fairly simple, which can be performed by typing make into the terminal as follows :

```
$ make
```

Compiler flag -O3 have been used for compiler optimization. Additional compiler flags can be specified by the user in the Makefile provided. The Makefile generates the executable `code` by linking `Thermal_Energy_Storage.c` and `function_list.c`. Subsequently, this executable can be run in the terminal to run the program as follows :

```
$ ./code
```

The program requires an integer input by the user to either continue with the default values specified in `user_input()` and `calc_param_sim()` functions (in this case, the user has to enter 0 in the terminal). Else, the user can enter any non-zero integer and enter each parameter himself.

```
$ Do you want to continue with the default values
(Enter 0) or enter the values yourself
(Enter a non-zero integer)
```

5.3 Interpretation of the code output

For visualizing the fluid and solid temperatures, the function `visualization` is called which prints these temperature values corresponding to a specific time instant (called as `time_current` in the program) into a `.dat` datafile, comprising of three columns. The first column is the non-dimensional position and other columns list the fluid and solid temperatures at these positions respectively. The datafile ends with five digits, specifying the time instant (in sec) at which the temperatures were captured. This is to prevent the duplicacy of the data files, with close time instants. The temperature profiles can be later visualized with Gnuplot (as used in the current work) by typing this into the terminal as follows :

```
$ gnuplot
```

This command takes you to the Gnuplot interface.

```
$ plot "filename.dat" using 1:2 w l, \
                "filename.dat" using 1:3 w l
```

This will plot the fluid and solid temperatures with default separate colors and lines. To save the current plot as an EPS figure, a Gnuplot plot file (with `.gplt` extension) can be used. User specific colors, line types, plot title, axis labels etc. can be directly specified into this plot file. (Sample Gnuplot plot files are available in the Folders where EPS figures have been stored). Then, one can directly call these plot files in the terminal as :

```
$ gnuplot filename.gplt
```

Other code output during MMS includes the printing of the error (absolute difference) between the numerically computed solution and the manufactured solution. Computation of this error is handled by the function `conv_error()`. Additionally, the function `error_compute_ovs_liquid()` and `error_compute_ovs_solid()` print the L_1, L_2 and L_∞ error norms into a datafile `error_ovs_fluid.dat`, along with the position of the maximum error. The same function also prints the absolute error of the temperature at every cell position `err_fluid_var.dat`, so that it can be easily plotted by the user to identify any possible mistake with the code.

The status of the storage medium is also printed into a datafile `status_storage.csv` at a certain time interval, so that the user can identify if the storage is running through all the charging, discharging and idle periods.

Chapter 6

Personal Experience

6.1 Lessons learnt from this Project

I consider this project **Numerical Simulation of a Thermocline Thermal Energy Storage** to be a stepping stone to enter the realm of *Computational Fluid Dynamics*. What makes me wonder is the amount of time and efforts needed to develop commercial softwares (ANSYS, COMSOL) which might be using curvilinear grids, unstructured meshes, multigrid methods and three-dimensional simulations, if this overly simplified one-dimensional two-phase flow problem consumed so much time!

Though I have taken a couple of related courses (Computational Fluid Dynamics, Applied Computational Methods in Mechanical Sciences and Numerical Methods) during my undergraduate studies with relevant course projects, I never experienced so much frustration when my codes seemed to generate errors. This is because, this project tested some fundamental topics like Grid refinement studies (Order of verification studies), Method of Manufactured solutions, code verification etc., which consumed enormous amount of time in coding and debugging, that were essential for verifying coding mistakes. Furthermore, I developed these codes using unfamiliar interfaces like : programming editor (Emacs) and plotting package (Gnuplot), which I had to learn along with the progress of the course project. Using the Linux terminal, Makefile (for compilation), LaTeXand git version control system (all these are the outcome of a course offered by *Computational Science and Engineering ETH - High Performance Computing in Science and Engineering I*) include other learnings as an outcome of this project. In short, this project enhanced my skills pertaining to these aforementioned packages and programming, in general.

From a mathematical perspective , I learnt a great deal about discretization, assessing the accuracy, order, convergence of the simulations, code verification, stability (in consistent with Von Neumann stability analysis) by applying the concepts which I had learned during the lectures. For instance, I realized my mistakes with the discretization and reconstruction approach while applying the Finite Volume Method, after discussions with the project supervisor.

Additional discussions cleared my misconception prevailing with the order of verification plots related to Peclet numbers. I also got an insight about the working of a thermal energy storage during the due course of this project (from the relevant temperature profiles plotted earlier). Additionally, this project also improved my writing skills by writing a report, which is essential for appropriately conveying the results obtained.
To conclude, my learnings from this project were multi-fold, and importantly it gave me an insight about the development of a CFD code, and its working. This rightly infers from one of the famous quotations in Computational Fluid dynamics by C. Hastings (1955) - *The purpose of computing is insight not numbers* !

6.2 Writing and testing the code from scratch

If one has to redo the project from scratch, the following steps in the development of code and in identifying the coding mistakes would be performed sequentially :

1. Understanding the background and objective of the work. This is essential to understand the physics behind the chosen model, so that one may realize if the numerically obtained results are consistent with the physics of the model.

2. After obtaining the governing equations of the physical model, it is essential to make some simplified assumptions (like one-dimensional, constant properties if the variation is insignificant etc.) to obtain the mathematical model (the one-dimensional two-phase flow problem in this case). In this stage, modeling errors might be introduced due to the simplifications.

3. Transforming the mathematical model to a discrete model, by discretization over a specific domain, so that the discretized equations can be solved over these discrete points. For this, identifying a suitable grid type (uniform in this case), grid arrangement (collocated in this case), and discretization method (spatial and time integration) will be necessary. In the current case, upwind and central differencing schemes for the advection and diffusion terms have been chosen for the spatial discretization. Additionally, explicit and point-implicit time-integration methods have been chosen.

4. To incorporate the variables and parameters to be used in the code, either according to the user-input or using the default values, dedicated functions will be written. Additionally, to visualize certain variables (temperatures in this case), functions can be written which print the temperatures of solid and fluid phases at specific time intervals into a datafile, which can be visualized in plotting packages. This can be tested by plotting the initial temperature input by the user.

5. In the scenario of the current project, the status of the storage can be determined by coding a similar function to print the storage status into a datafile. This can also be tested easily by visualization.

6. Since this is a two-phase flow problem, further simplifications may be made (h_v can be taken to be zero in this case) to reduce the coding mistakes as the project progresses. The discretized equations can be coded (with explicit time-integration in the first step) which solve the fluid and solid phase temperatures in separate functions. The code can be first developed to handle the charging period.

7. To test this initial code, one may resort to the Method of Manufactured Solutions, which is an essential method to identify the coding mistakes by numerically computing the order of accuracy. The simplest way is to choose trigonometric functions with a certain wavenumber which are continuous. For this, the aforementioned code developed to solve fluid and solid temperatures can be extended to handle these trigonometric functions and extra source terms if any. Plot the temperature profiles of the solid and fluid phases, and compare them to the exact profile (trigonometric variation) to determine if there is any significant deviation between the computed and exact solutions. It is equally necessary to plot the error between the fluid or solid temperatures and their respective exact solutions at each position across the length of the computational domain. If this is non-continuous or the variation of this error is not as expected, it may be wise to identify the points where this happens, which could be rectified in the code (Specifically double-check if the equation at the boundaries are right, as this is sometimes a common source of error). Once this has been done, the order of verification studies could be performed by varying the grid spacings and computing the respective L_1, L_2 and L_∞ errors. If the code is free of any errors, the actual order of accuracy should tend to the nominal order of accuracy, as the grid is refined in the asymptotic range of convergence. Any error obtained during this part clearly shows a mistake in solving the system of 2 equations or improper handling of explicit and implicit terms (source terms also).

8. Once the code is free of any errors for h_v equal to zero, the code may be extended to handle non-zero h_v which signifies the couping between the fluid and solid phase temperatures. In the current case, point implicit approach has been used by assuming the implicit variation of source terms only. Since the advection and diffusion terms are explicitly solved, a new function which calls the functions previously handling the explicit time-integration can be easily coded. Additionally, another function can be coded to solve the system of equations generated due to this implicit time-integration. Since the above code was developed only for the charging period, the code can be further extended to handle idle and discharging periods also. For this, a flag which knows the status of the storage can be used to determine the value of the interstitial fluid velocity. Additionally, boundary condition may have to be imposed on the right end of the

computational domain in the case of discharging period.

9. Once the code has been tested using the Method of Manufactured solutions, further code verification may be performed with an existing exact solution (if it exists) or validation with the experimental results. If there is significant deviations between the plots, errors in coding the formulations of h_v, $h_{f,s}$, Pr, Nu or $u_{f,i}$ might be possible. Capturing the temperature profile at a miscalculated time instant could also be a reason for the deviation. Once the numerical results are in good agreement with the exact solution, then the code is ready for performing full scale simulations.

10. Computation of capacity factor, cycle exergy efficiency and temperature increase at outflow may be necessary to determine the reliability of the storage medium of different geometries. So, these can be incorporated through dedicated user defined functions.

Bibliography

[1] A. Haselbacher, 2017.

[2] P. Knupp and K. Salari, *Verification of computer codes in computational science and engineering*, ser. Discrete mathematics and its applications. Boca Raton, Fla: Chapman and Hall, CRC, 2003.

[3] G. Zanganeh, A. Pedretti, S. Zavattoni, M. Barbato, and A. Steinfeld, "Packed-bed thermal storage for concentrated solar power – pilot-scale demonstration and industrial-scale design," *Solar Energy*, vol. 86, no. 10, pp. 3084 – 3098, 2012.

[4] B. M. P. A. Z. G. Zavattoni, S.A., "Cfd simulations of a pebble bed thermal energy storage system accounting for porosity radiation effects," ser. SolarPACES.

[5] G. Zanganeh, "High-temperature thermal energy storage for concentrated solar power with air as heat transfer fluid," Ph.D. dissertation, ETH Zurich, 2014.

[6] B. P. Leonard, "Stability of explicit advection schemes. the balance point location rule," *International Journal for Numerical Methods in Fluids*, vol. 38, no. 5, pp. 471–514, 2002.

[7] E. Krause, "Ferziger, j. h.; perić, m.: Computational methods for fluid dynamics. berlin etc., springer-verlag 1996. xiv, 356 pp., dm 74,00. isbn 3-540-59434-5," *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 77, no. 2, pp. 160–160, 1997.

[8] S. Xue, N. Phan-Thien, and R. Tanner, "Upwinding with deferred correction (updc): An effective implementation of higher-order convection schemes for implicit finite volume methods," vol. 108, pp. 1–24, 12 2002.

[9] S. Srivastava, "Operator splitting methods for the advection-diffusion-reaction equation," 2008.

[10] H. P. Langtangen and S. Linge, "Finite difference computing with pdes."

[11] D. Lanser and J. Verwer, "Analysis of operator splitting for advection–diffusion–reaction problems from air pollution modelling,"

Journal of Computational and Applied Mathematics, vol. 111, no. 1, pp. 201 – 216, 1999.

[12] "Intel® xeon® processor e5-2697 v2 (30m cache, 2.70 ghz) product specifications."

[13] Wikipedia, "Gaussian elimination — wikipedia, the free encyclopedia," 2017, [Online; accessed 26-November-2017].