Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Materials Science and Engineering

Fatbardha Hajdini

Dynamic Modelling of Heat and Mass Transfer in the Silicon Arc Furnace

Master's thesis in Chemical Engineering and Biotechnology
Supervisor: Kristian Etienne Einarsrud.
Co-supervisors: Halvard Tveit and Leiv Kolbeinsen
July 2019

NTNU
Norwegian University of
Science and Technology

Fatbardha Hajdini

Dynamic Modelling of Heat and Mass Transfer in the Silicon Arc Furnace

Master's thesis in Chemical Engineering and Biotechnology
Supervisor: Kristian Etienne Einarsrud.
Co-supervisors: Halvard Tveit and Leiv Kolbeinsen
July 2019

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Materials Science and Engineering

NTNU
Norwegian University of
Science and Technology

Acknowledgment

First of all I would like to thank my supervisor Kristian Einarsrud and my co-supervisors Halvard Tveit and Leiv Kolbeinsen for their support. An extra thanks to Halvard for the constructive coversations during the last stretch of the thesis.

The current work has been organized in HighEFF - Centre for an Energy Efficient and Competitive Industry for the Future, an 8 year Research Centre under the FME-scheme (Centre for Environment-friendly Energy Research, 257632/E20). Support from the center is acknowledged.

Finally I would like to thank my family for showing their support and always taking care off me. I would also like to thank my friends, both far and close, but especially those I shared my reading room with, for the jokes and long talks and very pleasant lunches. The master year would not have been as pleasant without you! A special thanks to my little sister Gjylsime for listening to me on daily basis. A special thanks to Haakon for his love and support.

Abstract

The element silicon is of high importance for the modern human society. Although in high abundance, no silicon is in elementary form in the nature. Metallurgical grade silicon can be produced in a submerged arc furnace. The furnace can be divided into two zones: a low temperature zone where the raw materials, coal and quartz, are added and is located above the cavity around the electrodes, and a high temperature zone which consists of a cavity, electrode and molten silicon bath. The silicon process consumes large amounts of electrical energy and generates large amounts of waste heat; both from the off-gas and molten silicon leaving the furnace at elevated temperatures. The objective with this work is to use an open source platform (OpenFOAM) to make a framework of both the low temperature zone and the high temperature zone in silicon process. By using dynamic models that consider both heat and mass flows, the waste energy can be better utilized and subsequently reduce the total energy consumption of the process.

The models aim at calculating dynamic temperature and concentration profiles capturing the effects of heat transfer between the gas and solid phase, and heat and mass production and consumption due to the reactions inside the furnace. The effect of change in carbon reactivity and carbon concentration in the low temperature model were investigated. In the high temperature model the effect of change in SiC(s) concentration was investigated, showing that the model could capture some changes due to the changes in these parameters. The models of the low and high temperature zone worked, however the framework need to be further developed in order to sufficiently describe the silicon arc furnace sufficiently. The framework managed to calculate the cases rather fast, and it is shown that information can be taken out of each sub-model and used used in the other in order to couple the two models dynamically in future work.

Sammendrag

Grunstoffet silisium er av stor betydning for det moderne samfunn. Selv om silisium er det vanligtse grunnstoffet i jordskorpen etter oksygen, så finnes ikke silisium i sin elementære form i naturen. Silisium kan produseres i en nedsenket lysbueovn. Ovnen kan deles inn i to soner: en lavtemperatursone der råmaterialene, kull og kvarts er tilsatt og ligger over hulrommet rundt elektrodene, og en høytemperatursone som består av et hulrom, elektrode og smeltet silisiumbad. Silisiumprosessen forbruker store mengder elektrisk energi og genererer store mengder avfallsvarme; både fra avgassen og smeltet silisium som forlater ovnen ved forhøyede temperaturer. Målet med dette arbeidet er å bruke en åpen kildekodeplattform (OpenFOAM) for å lage et rammeverk for både lavtemperatursonen og høytemperatursonen i silisiumprosessen. Ved hjelp av dynamiske modeller av varme- og massestrømmene i ovnen, kan spillvarme bli bedre utnyttet, dermed kan det totale energiforbruket i prosessen reduseres.

Målet med modellene er å beregne dynamiske temperatur- og konsentrasjonsprofiler som fanger effekten av varmeoverføring mellom gass og faststoff, samt varme og masseproduksjon og -forbruk på grunn av reaksjonene i ovnen. Virkningen av endring i karbonreaktivitet og karbonkonsentrasjon i lavtemperaturmodellen ble undersøkt. I høytemperaturmodellen ble virkningen av endring i SiC(s) konsentrasjon undersøkt, noe som resulterte i at modellen kunne fange noen endringer på grunn av endringene i disse parametrene. Modeller av lav- og høytemperatursonen fungerer, men må videreutvikles for å tilstrekkelig beskrive silisiumovnen i tilstrekkelig grad. Rammeverket klarte å beregne de ulike casene ganske fort, og viste at informasjon kan tas ut av hver delmodell. Dette kan i fremtiden brukes til å mate informasjon fra en delmodel til en annen og koble de to modellene dynamisk.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Silicon as an element is found in high quantities in the earth's crust, second only to oxygen [1]. In nature it is almost exclusively combined with oxygen as a fairly pure silicon dioxide and silicates [1], from which is can be extracted. As early as the the stone age, silicon dioxide (as flint) was an important tool. Later, silicates have been essential in materials such as ceramics and glass. Early in nineteenth century silicon was recognized as an element and produced on industrial scale as a element or in alloys from the end of the century. Important properties of silicon as an metalloid are its extreme bonding qualities and easy doping possibilities [2]. Separating silicon and oxide in the SiO_2-formation is the main challenge in producing metallic grade silicon [2]. The bonding between Si and O is very stable due to the ionic size of silicon, which needs to share four electrons in the 3rd electron shell, but can only share one electron with each oxygen atom. Before the computer industry and the naming of Silicon Valley became a symbol of the modern world in 1971, silicon and its alloys were rather unknown to the public [1]. Silicon is now an important part of the daily life of most human beings on Earth [2], even if they are aware or not. The development of usage of silicon has been spectacular both is use of silicones and in computer chips.

According to Tveit and Kolbeinsen [2] silicon will play an important role in solving the challenges that arise due to the soon 8 billion people in the human society:

1. Possible constraints in future supply of energy
2. Increased demand of energy
3. Environmental consequences of production and usage of some of the main energy resources.

Silicon contribution to energy production will be in solar cells and possible in the important battery developments. Silicon is also used in the production of lightweight materials used in transport, as well as in silicones and computer development.

1.1 The silicon process

Silicon is commercially produced in a submerged arc furnace by carbothermic reduction of quartz [1, 3, 4, 5, 6, 7], which requires high temperatures and is an energy intensive process [8]. According to Takla et al. [8] and Schei et al. [1] most plants use 11-13 kWh of electrical energy per kilogram of silicon produced, where the furnace is the most energy consuming part of the plant [6]. The electrical energy represents about 45% of the total energy supplied to the process, while the rest is supplied by the raw materials [1, 8]. Energy balances of the process show that roughly 70% of the total energy input leaves the process as thermal energy in cooling water, by radiation and convection from the furnace, as hot off-gasses in addition to the cooling process of the liquid silicon; the remainder 30% of the total energy input is contained in the product silicon [1]. In order to improve overall process resource utilization, thermal energy is an untapped energy resource, which the industry is well aware of [8]. Several plants have installed some sort of energy recovery equipment, others have investigated the possibilities for better waste heat utilization through industrial clusters and industry parks. The international energy consumption of silicon production in 2017 can be calculated to be around 100 TW [1, 9].

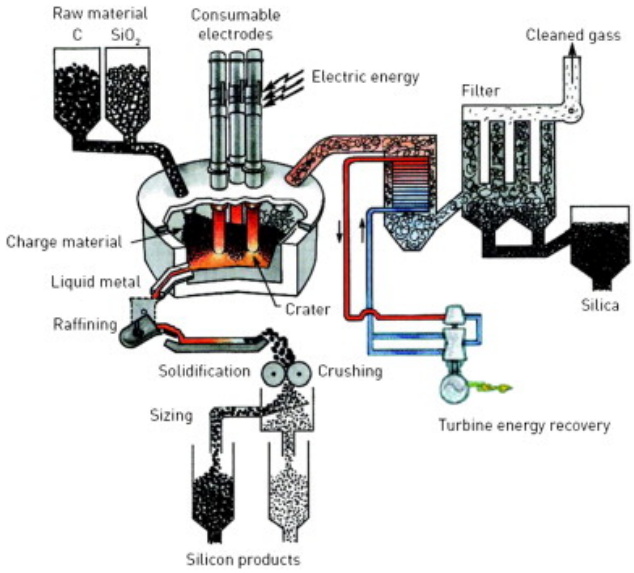


Figure 1.1: Overview of a silicon plant. Taken from Schei et al. [1].

The electric system in a furnace consists of three electrodes, symmetrically arranged and conducting electrical energy to the hearth of the furnace. They are submerged into the charge, supplying enough energy into the hearth to run the endothermic reactions [3, 10]. Around the electrodes there are formations of cavities, in which the current forms electric arcs, as a result of extensive heat generation in the charge around the electrode tip. The electric arc heats up parts of the charge to about 2000 °C in the hottest parts of the furnace [1, 10]. When the charge has reached this temperature, silicon dioxide is reduced to molten silicon and drained either continuously or discontinuously from tap-holes at the bottom of the furnace. The molten silicon is then sent to refining. The aim during the refining process is to remove particles of oxide and carbide, and composition is adjusted to the grade of metal wished to be produced [1, 6]. After this process the molten silicon can cool in a suitable mould before it is crushed into specific sizes.

During the production of silicon, $\mathrm{SiO(g)}$ and $\mathrm{CO(g)}$ is produced. According to Schei et al. [1] the gas leaving the furnace average temperature of the gas leaving the top of the furnace have a measured temperature of 1400 °C. This is sent to gas cleaning facilities, where the gaseous products burn with air

from the surroundings, and microsilica ($SiO_2(s)$) and carbon dioxide ($CO_2(g)$) is formed. Microsilica is recovered from the waste gas and the gas is cleaned.

1.2 Chemical reactions occurring inside the furnace charge

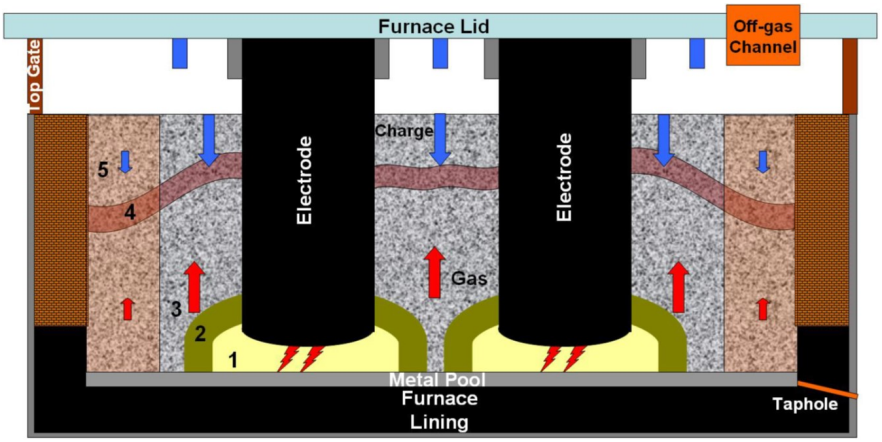


Figure 1.2: A cut through, graphical depiction of the submerged arc furnace used in high silicon alloys production. Different zones have been formed in the charge such as crater zone (1), crater wall (2), softening and melting zone (3), crust formation zone (4) and stagnant charge zone (5). Taken from Kadkhodabeigi [3].

In order to produce metal in the submerged arc furnace trough carbothermic reduction of the ores, several chemical reactions are needed. The general chemical reactions occurring in a silicon producing furnace are as follows according to Schei et al. [1]:

$$SiO_2(s,l) + 2\,C(s) \longrightarrow Si(l) + 2\,CO(g), \quad \Delta H{=}870.21\,kJ/mol \tag{1.1}$$

$$SiO_2(l) + C(s) \longrightarrow SiO(g) + CO(g), \quad \Delta H = 792.510\,kJ/mol \tag{1.2}$$

Reaction 1.1 and 1.2 are composed of different chemical reactions occurring in the charge. The endothermic nature of the main reactions show that the heat distribution in the charge material plays an important role in determining the rate of reaction in different zones in the furnace. The insides of the furnace is generally considered as two parts, known as the inner (zones 1, 2, 3 and lower part of zone 5) and outer parts (zone 4 and upper parts of zone 5) shown in figure 1.2, where the inner parts are the active parts of the furnace. In the inner part of the furnace the main reactions are (Schei et al. [1]):

$$SiO_2\,(s,l) + Si(l) = 2\,SiO(g), \quad \Delta H = 605.441\,kJ/mol \tag{1.3}$$

$$SiO\,(g) + SiC\,(s) = 2\,Si\,(l) + CO(g), \quad \Delta H = 166.71\,kJ/mol \tag{1.4}$$

$$SiO_2\,(s,l) + SiC\,(s) = 3\,SiO\,(g) + CO(g), \quad \Delta H = 1377.597\,kJ/mol \tag{1.5}$$

Carbon is only available in solid phase as carbon has a very high melting point, and is only attending the reactions in solid phase in reaction with liquid and gas phases available in the furnace. On the other hand, quartz has a lower melting point and is softening in the middle part of zone 3 in figure 1.2. The melting occurs close to the electrode in boundaries of zone 2 and 3 shown in the same figure. The silica melt then flows toward the furnace crater and causes reaction (1.3) to happen. Reaction (1.3) is the most important SiO forming reaction in the system, and needs high amounts of energy in order to run. This reaction occurs in the

high temperature zone of the furnace, under the electrode tips marked as zone 1 in figure 1.2. Reaction (1.4) is the most important metal producing reaction, which is strongly endothermic and requires high amounts of energy in addition to an environment with high partial pressures of SiO gas in order to occur. This reaction probably occurs in zone 2. The produced gasses in the inner zone during the process contain SiO(g) and CO(g), which flows towards the furnace top through the charge materials. The formation of SiC in reaction (1.6) is one of the most important reactions in the outer zone according to Schei et al. [1]:

$$\mathrm{SiO\,(g)} + 2\,\mathrm{C\,(s)} \xrightarrow{\mathrm{R_1}} \mathrm{SiC\,(s)} + \mathrm{CO(g)}, \quad \Delta\mathrm{H} = -77.592\,\mathrm{kJ/mol} \tag{1.6}$$

This reaction plays an important role in recovery of silicon in the furnace and hence results in higher production yield. This reaction mostly occurs in zone 3 shown in figure 1.2. Physicochemical properties of the carbonaceos materials used in the charge (such as particle size and reactivity) are important, as they affect the flow of reactive gas in the bed of solid particles in this zone, and proper flow is very important in determining the rate of gas-solid reactions [3]. Condensation of SiO gas is another phenomena which happens in the upper part of the furnace charge (zone 4 in figure 1.2). This reaction occurs when the charge temperature is less than the condensation temperature. Condensation based on reaction (1.7) reduces the escape of SiO gas from the furnace, hence increase silicon recovery in the furnace.

$$2\,\mathrm{SiO\,(g)} = \mathrm{SiO_2(s,l)} + \mathrm{Si(l)}, \quad \Delta\mathrm{H} = -306\,\mathrm{kJ/mol} \tag{1.7}$$

According to Schei et al. [1] the product of reaction (1.7) is found in a relatively thick layer of brown substance in the upper part of the charge.

1.2.1 Formation of different zones

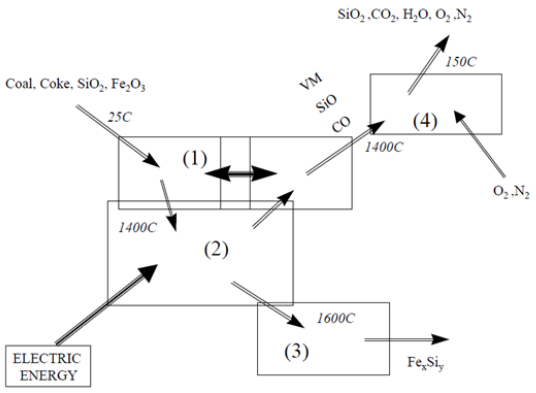


Figure 1.3: The principal process flow in the ferrosilicon production process. Taken from Schei et al. [1].

The principles of the different zones in the silicon process is shown i figure 1.3. It is the zone (1) and zone (2) that will be modelled in this work. The zone (3) - silicon solidification and zone (4) - off gas combustion is outside the scope of this thesis.

1.3 Process requirements of raw the materials

Quartz and carbonaceous materials as reductant are the main raw materials used in the silicon production [1]. There are quality requirements for these raw materials in which can be divided into following groups:

- Process
- Product
- Environmental issues

Quartz and quartzite are used as siliconoxide source for silicon and ferrosilicon production. Generally, quartz is used in silicon production due to higher need of purity. The process requirements of quartz and quartzite are related to size and strength. Thermal strength of quartz is an important parameter since it can affect the gas flow in the charge if too much quartzite fines are present. It is not favorable for the process operation if the the quartzite melts too high up in the furnace. Therefore some silicon producers use thermal tests for quartz and quartzite. Undersized material is normally unwanted as it also can affect the gas flow in the charge and may be sieved and removed before being fed to the furnace. The normal size of quartzite varies between 10 to 150 mm.

It is commonly considered that the quality of the reduction materials is important to achieve a high silicon yield in the furnace [1]. Both the size and reactivity of the carbon that is used will have an effect on the process performance. The size of the carbon varies from 1 up to 30 mm. Decreasing size of the raw material will increase the specific surface. The small particles in the furnace may be carried out into the gas outlet due to the high gas velocity in the furnace. This might lead to losing control over the amount of carbon to the process. According to Johansen et al. [11] the critical minimum size of charcoal particles can be up to 1.5 mm to avoid this.

One of the most important process parameters according to Tuset [12] is the ability of a carbon source to react with SiO gas from the crater according to the reaction:

$$\mathrm{SiO\,(g)} + 2\,\mathrm{C(s)}{=}\mathrm{SiC\,(s)} + \mathrm{CO\,(g)} \tag{1.8}$$

the reaction of carbon according to equation (1.8) is highly dependent on the carbon quality. This has been shown in both practical experiments and laboratory measurements. There are several tests for carbon material in silicon production that are in use. The SINTEF test uses the reaction (1.8) as a reactivity criterion.

1.4 Over and under coked

According to Schei et al. [1], the carbon content of the charge is one of the most important parameters of furnace operation. These are one of the parameters the metallurgist can easily vary. The best furnace operation today normally do not give severe problems with overcoking. But earlier the furnaces could be totally clogged with SiC and had be excavated or totally rebuilt, often after a short time period. In an example in Schei et al. [1], the dynamic model of Elkem is used to simulate a furnace going from a stable period to one week overcoked and one week undercoked. In the stable period the Si recovery is 85 percent. When C content is increased the Si-recovery drops for some hours. This is due to the reaction rate increase in SiC production with increased C content in the charge, while the consumption of SiC is proportional to the amount of SiC and therefore not changed in the beginning.

Decreasing the carbon content, the rate of SiC consumption is high after one week because of the available SiC deposits, while production of SiC drops. The deposits of SiC will be converted to Si, hence a sharp

increase in the Si recovery. As the stores of SiC(s) is consumed however, the rate of produced Si(l) decreases and approaches a stationary value corresponding to the new charge composition. Therefore the Si recovery has to drop after some time.

1.5 Significance

The complexity of the silicon process enhances the need for understanding of the most important parameters to secure optimal operation. A successful model of the process may give better prediction of the energy and mass balance in order to reduce the energy consumption and improve the energy standard of the process. The model may be used to study the high and low temperature that is difficult or impossible to measure. This could be used to predict the temperature of the off-gas leaving the furnace in order to better utilize the waste energy.

1.6 Objectives and scope

The aim of this thesis is to demonstrate simulation of the silicon arc furnace by dividing the furnace into relatively simple building blocks to describe the essential features of the Si-process. I will describe essential features of high and low temperature zones and implement models for each of them. The goal of the models is to dynamically describe the temperature and concentration profiles, which are affected by reaction kinetics, heat transfer (between the gas and solid phase), and convection. Five main chemical reactions will be included in the model in order to investigate their effect on the concentration and temperature profile.

Due to time limitations the main focus of the thesis will be on modelling and simulations of the low and high temperature zone in addition to verification of the models, even though experimental measurements are crucial in order to validate the model, that is not focus in this thesis.
The main objectives of this thesis are:

- To present a review of some current models of the silicon arc furnace
- To develop simple and dynamic frameworks for the low and high temperature zone in the silicon arc furnace separating the solid and gas phase in order to obtain more detailed solid-gas temperatures and concentration profiles.
- To verify the essential features of the simulation framework.
- To investigate how different parameters affect the dynamics of the modeled zones in the furnace.

Chapter 2 contains a literature review of some existing models. In chapter 3 the theoretical framework and system equations used for the two models created in this work are presented. Theory about Computational Fluid Dynamics and `OpenFOAM` is presented in chapter 4. Parameter determination for the framework is performed in chapter 5, and verification of the models is done in chapter 6. Results from parameter studies is presented in chapter 7. The results of this work are discussed in chapter 8. Chapter 9 includes the conclusions and ideas for further work.

Chapter 2

Literature Review

There have been several attempts in creating models to describe the silicon process or parts of the silicon furnace in order to understand it and reach an optimal furnace operation. The most relevant models for this work are presented in this chapter.

2.1 The Elkem models

Both a stoichiometric and a unidimensional dynamic model of the silicon process has been developed by Elkem. These are both described in several publications. The stoichimetric model has been described by Schei et al. [1] and in Schei and Halvorsen [13]. The stoichimetric model is a simple steady state model that captures the main reactions reactions between furnace compounds, calculating arithmetic balances between the species in order to determine how much of each material can be found in an inner zone at the base of the furnace and an outer zone in the furnace top. The model combines equilibrium and kinetic considerations with material and energy balance. The model is implemented in Microsoft Excel and is based on furnace observations and knowledge of the chemistry of the system. One of the main advantages of this model it that a case can be calculated in a few seconds on a personal computer, but it can still show many of the characteristic properties of the silicon process. However, a shortcoming of the model is that it only applies to steady state operation and does not show how the system approached the steady state. The stoichimetric model uses lumped parameters, each parameter account for several physical effects, so the parameters are not accessible for independent measurements.

The dynamic model developed by Halvorsen et al. [14] and also described by Schei et al. [1] is a unidimensional dynamic model of the silicon process for the metallurgical behaviour of the upper part of the furnace [1], describing the state as a function of height in the furnace. The furnace is split into two main parts: the hearth where the electric power is supplied and a shaft which is segmented to get a reasonable resolution of variation in the vertical direction. The model described gas composition, reaction rates as well as transport of materials and energy between the segments of the shaft. The calculations of a case can take something between minutes and hours on a personal computer. The parameters in the shaft can in principle be measured independently and have real physical meaning. The model can switch between two modes: a continuous feeding mode and a stoking mode. In the continuous feeding mode, the model describes the approach to steady state for a set of parameters. These are suitable to studying long-term development of the process. The stoking mode should should be used when variations within a stoking cycle are important.

2.2 Modeling of tapping processes in submerged arc furnaces

Mehdi Kadkhodabeigi [3] used both CFD modeling techniques and industrial tests to investigate three main topics:

1. Tapping process in silicon and ferrosilicon production furnace
2. Designing a new hood system for capturing of tapping off-gases and ladle fumes
3. The tapping process in ferromanganese furnaces

Only the first and last topics are presented in this section.

2.2.1 Tapping process in silicon and ferrosilicon production furnace

A 3D multi-phase CFD model of a silicon and ferrosilicon process was developed by Kadkhodabeigi [3]. The model of the furnace was based on industrial geometry and the most probable conditions inside the furnace. The model is dynamic and was used, among other things, to study the effect of different parameters on the tapping speed. A study of a full 3D muliphase models based in comptational fluid dynamic, interactions between the different phases and fluid flow of the silicon and ferrosilicon process were developed. Due to the huge size of the furnace model, obtaining the results from the developed model it this work were very expensive from a computational point of view.

2.2.2 The tapping process in ferromanganese furnaces

In order to study the tapping process of a ferromanganese furnace Kadkhodabeigi [3] also developed a 3D multi phase model of the high temperature zone. The focus of the model is on the melt flow inside the furnace during the tapping process, in addition to temperature distributions in the melts and furnace refractory walls. The initial conditions of the inside of the furnace were modeled based on the most probable situations which he managed to find from literature review or industrial data. The model geometry was based on industrial size furnaces. Industrial data of the tapping weight of metal were compared to the model output in order to find a range for the metal height in the furnace. However, there is no industrial measurements of metal height in the furnace so this could not be compared. The model showed distinct temperature differences between slag and metal in the furnace, so the results from the model show that it might be possible to estimate volume flow of slag and metal during taping through temperature control in melt streams.

2.3 A heat and mass transfer model of a silicon pilot furnace

Sloman et al. [5] developed a mathematical model of a silicon arc furnace. A continuum approach was taken, derived from first principles equations governing the time evolution of chemical concentration, gas partial pressure, velocity, heat transfer and temperature in a one dimensional vertical section of a furnace. They assumed instantaneous heat transfer between gas and the solid and liquid and graphite crucible, using only one temperature. The model was applied to silicon pilot furnaces in order to gain better understanding of silicon pilot furnace experiments. In order to understand crust formation within the furnace, numerical simulations replicated the reactions of the lower part of the furnace, corresponding to the creation of a gas cavity and crater region. The result were compared with experimental results and show a good fit for the position of the interface between the charge and the lower regions. This was done without the need to fit any model parameters artificially.

2.4 FFI

According to Schei et al. [1], the creation of a very advanced model of the hot zone of the silicon furnace was sponsored by the Norwegian Ferroalloy Producers Research Association (Norwegian: *Ferrolegeringsindustriens Forskningforening*, FFF). The two dimensional model describes the situation around the tip of the electrode which consists of a gas filled cavity around the electrode tip, the metal pool underneath, and the lower part of the charge above the cavity. The geometry of the model is sylindrical containing one inert electrode with the center as the axis of symmetry. Heterogeneous reactions at temperatures above 1860 °C, evaporation and condensation of silicon, transport of materials by dripping, direct current electric arc, turbulent and laminar flow, heat transfer by conduction, convection and radiation are the most important phenomena included in the model. One of the disadvantages of this model is the size, it is too complex for a personal computer. Computing times for some of the largest computers is about several hours. The FFF model is however suitable for throughout studies of the heath of the furnace with some improvements.

2.5 Model comparison

According to Schei et al. [1] it is expected that a universal model probably never will be developed, as such a program would be so difficult and time consuming to run that it would be better to perform direct measurements on real furnaces instead. However, various purposes need various models, and in order to create a useful model simplifications are required, while staying accurate with regards to the properties the model is supposed to describe. The models presented in this chapter are modeled in such a way that the simplifications, assumptions, model domain, numbers of phases etc. that are accounted for are fitted to the aim of the study. In order to compare these model a short comparison is presented in table 2.1. The aim of the model in this work is to stand out from the others by the focus of modelling the temperature of the gas and solid phase separately in order to predict the temperature of the gas leaving the furnace to better utilize the waste heat due to elevated temperatures in the gas phase.

Table 2.1: Comparison of models presented in literature review. “Current model” is the model developed in this thesis.

Description \ Model	Current model	Stoichiometric [1]	Dynamic [1]	FFF [1]	Kadkhodabeigi [3]	Sloman et al.[5]
Dimensions	1D	-	1D	2D	3D	1D
Chemical reactions	yes	yes	yes	yes	yes	yes
Temperatures	Two	one	one	one	one	one
Model volume	High and low temp. zone	Overall furnace	Shaft and hearth	High T.area	Overall furnace	Overall furnace
Coordinate system	Cartesian	-	Cartesian	Cylindrical	Cartesian	Cartesian

Chapter 3

Model Development

Simulations can be a useful tool in gaining improved understanding of the silicon process[15]. In this chapter, I present the general approach to modelling both the low and the high temperature zone of the active area of a silicon arc furnace. The aim of this chapter is to present the system equations and theories used in the framework. I implement several functions in order to capture some of the physics inside the furnace: Electrode heat generation, reaction enthalpy calculations, heat transfer coefficient and reaction rate calculations. Each will be presented in separate sections in this chapter.The chapter is divided into two main sections: The low temperature model (LT model) and the high temperature model (HT model) in sections 3.1-3.7.1 and 3.8 respectively. The two models have many similarities, these will not be be repeated in 3.8 only mentioned in the introduction of the model. The programming language will be OpenFOAM, some parameters needed are calculated in Microsoft Excel or MATLAB. Many of the reaction rates necessary for model were not obtained. This combined with the complexity of the simulated process created a huge amount of necessary tuning. The initial model was extremely slow (17 hours simulation time for 1 sec real time simulation). In order to finalize the model and obtain results the model had to simplified several times. The complete code for the base cases can be found in the Appendix.

3.1 Low temperature model

In this section, a 1-D mathematical model of the low temperature zone of a silicon furnace is developed in order to simulate the behaviour of this zone. This low temperature model is placed in the active area of in zone 1 in figure 1.3. The section will start by indicating which chemical species and reactions will be considered and choice of geometry. Then the framework of the model is formulated by considering conservation of the chosen chemical species and energy. The gas is assumed to behave ideally and the total gas pressure used to calculate partial pressures is held constant at 1 atm. In this thesis the bracketed notation (s) denotes solid, (l) liquid, and (g) a gas. The system is a porous medium, containing gas phase, solid phase and a liquid phase. In order to maintain conservation of energy in the system, it is divided into two phases: a joined solid-liquid phase and a gas phase. The conservation of energy will be written in terms of temperature of the gas and joined solid-liquid phase, which from now on will be denoted the solid phase. In order to derive equations to describe the fluid and solid flow of the species in our systems, governing equations and simplifications of these are needed. These are derived from first principles such as conservation of mass and energy. The one-dimensional models can be naturally extended to 2-D or full 3-D geometries, if conservation of momentum is included [5]. Figure 3.1 shows a sketch of the low temperature model with inlet temperature

from the high temperature zone, inlet temperature and concentration of the charge, and height of the system. It is assumed that the temperature of the gas leaving the hearth is 2000°C and that the raw materials are fed with room temperature of 25°C to the furnace. Calculations of concentration of the gas species and raw materials are explained in chapter 5.

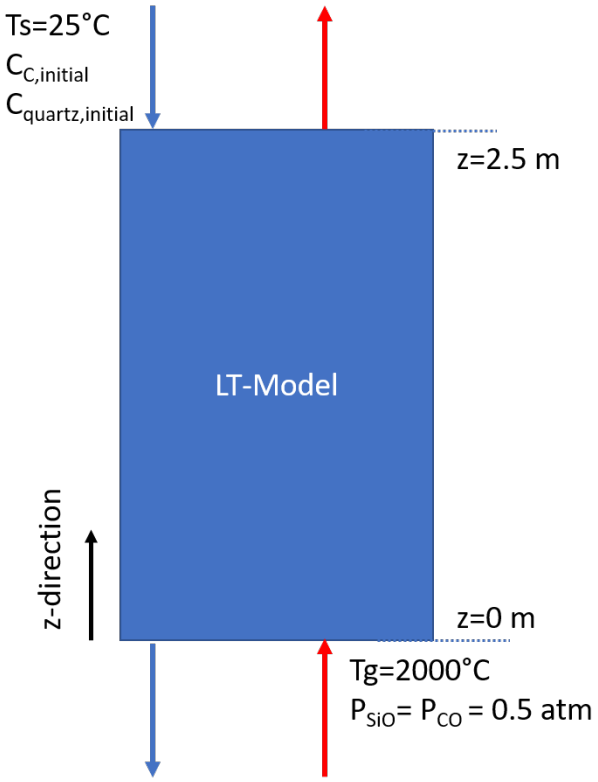


Figure 3.1: The principle flow chart of the low temperature model created in this thesis.

3.2 Geometry

For simplicity, a 1-D vertical section of the low temperature zone in the active area of the charge is modeled. By letting z denote the zone height, we take base at $z = 0$, and top $z = h$ as illustrated in figure 3.1. We are interested in the temperature and concentration profiles inside the furnace and in each zone. We therefore model this zone as a two-phased porous model. The height of the charge was set to 2.5 m, in accordance to normal furnace dimensions [3]. The mesh consists of 100 quadratic cells. Mesh is defined in section 4.1.

3.3 Chemical species and reactions

According to Andresen [16], C(s), SiC(s), SiO_2(l, s), Si(s, l) in addition to condensates of SiO_2, Si and possibly some SiC(s) and C(s) are the chemical compounds believed to be important for the metallurgy of the carbothermic silicon metal production. SiO(g) and CO (g) are produced or consumed in heterogeneous chemical reactions inside the furnace. Si(g) is produced by evaporation of Si(l), chemical reactions or dissociation at high temperatures. In this thesis production of Si(g) is neglected, in addition to condensation of Si(s); it is assumed that Si is only present in liquid state. Hence, in this thesis we are interested in modeling the following species:

- C(s) - carbon
- SiC(s) - silicon carbide;
- SiO_2(s) - silica, solid quartz or quartzite
- SiO_2(l) - liquid 'sticky' silica;
- Si(l) - liquid silicon condensate;
- CO(g) - carbon monoxide;
- SiO(g) - silicon monoxide

Other compounds such as the impurities Al, Ca and their oxides and Ti can also be found in small quantities in industrial furnaces and may contribute to the dynamics [1, 5], but will not be taken into account as they are outside of the scope of this thesis. When discussing the chemistry of silicon metal production, non volatile components other than Si, C and O (Al_2O_3, CaO, Fe_2O_3, TiO_2) can be disregarded as the chemical species used in the process are quite pure [16].

There are several reactions that can occur in the silicon furnace as presented in section 1.2. According to Schei et al. [1] only five of these are actually needed to describe the thermodynamics of the system. This is when assuming that SiO and CO are the only gas components that matter under the prevailing conditions. In addition to this, we have the melting of quartz. The mentioned reactions are [16, 17, 5]:

$$\mathrm{SiO(g)} + 2\,\mathrm{C(s)} \xrightarrow{\mathrm{R_1}} \mathrm{SiC(s)} + \mathrm{CO(g)}, \tag{3.1}$$

$$2\,\mathrm{SiO(g)} \underset{\mathrm{R_{-2}}}{\overset{\mathrm{R_2}}{\rightleftharpoons}} \mathrm{SiO_2(l)} + \mathrm{Si(l)}, \tag{3.2}$$

$$\mathrm{SiO_2(s)} \xrightarrow{\mathrm{R_3}} \mathrm{SiO_2(l)}. \tag{3.3}$$

$$\mathrm{SiO(g)} + \mathrm{SiC(s)} \xrightarrow{\mathrm{R_4}} 2\,\mathrm{Si(l)} + \mathrm{CO(g)}, \tag{3.4}$$

$$\mathrm{SiO_2(l)} + \mathrm{C(s)} \xrightarrow{\mathrm{R_5}} \mathrm{SiO(g)} + \mathrm{CO(g)}, \tag{3.5}$$

$$\mathrm{SiO_2(l)} + \mathrm{SiC(s)} \xrightarrow{\mathrm{R_6}} 6\,\mathrm{SiO(g)} + \mathrm{CO(g)}, \tag{3.6}$$

Reaction (3.6) is excluded in this model, just as [16] and [5] did in their modeling work. In order to limit the work done in this thesis the back reaction in equation (3.2) was neglected.

The chemical species are treated in such a manner that they exist in one of either a joint solid and liquid bulk phase or a gas phase. Chemical compositions will be calculated in concentrations, using notation C_i for the concentration of species i with units mol/m^3. Concentration, $C_{SiO_2(l)}$ is the number of moles of SiO_2(l) per unit volume of the joined solid-liquid bulk phase. While the concentration of any gas species such as C_{CO} is the concentration of species CO(g) in the volume the gas phase exists in.

3.4 Fluid flow through a porous medium

According to Nield and Bejan [18], a *porous medium* is a material that consists of a solid matrix with an interconnected void. Here we assume that the solid matrix is rigid, even though there are formations of liquids in the model. It is the interconnection of the void that allows for gas flow though the material [18]. A schematic view of a solid porous bed in presented in figure 3.2.

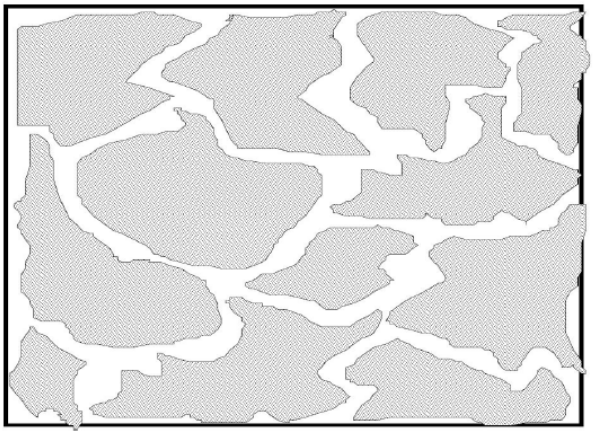


Figure 3.2: Schematic view of a solid porous bed. The white color represents the void fraction, and the grey color represents the solid particles. The picture is taken from [3].

The porosity of a porous medium,φ, is defined as the fraction of the total volume of the medium that is occupied by void space [18]. Hence (1-φ) is the fraction that is occupied by the solid. Asssuming an isotropic medium, the surface porosity, the fraction of void area to total area of a typical cross section, of equal to φ. Normally φ does not exceed 0.6 for natural media, and for beds of natural spheres φ can vary between the limits 0.2505 and 0.4764. When grain sizes are non uniform, the porosity tends to be smaller. For this model the porosity was set to 0.3 in accordance with Tveit [19].

3.4.1 Velocity of gas through a porous medium

When working with fluid flow though porous medium it is important to distinguish between intrinsic average velocity, $v_f^{'}$, and superficial velocity, v_f, of the fluid flow. The intrinsic fluid velocity is calculated taking an average of the fluid velocity over the volume occupied by the fluid, also seen as the fluid flux [20]. The superficial velocity, also called seepage velocity or Darcy velocity, is calculated over the total volume of the medium V_m (both solid and fluid material) [18]. The intrinsic fluid velocity is related to the superficial velocity by the Dupuit-Forchheimer relationship:

$$v_f = \varphi v_f^{'} \tag{3.7}$$

In classical fluid mechanics void fraction $\varphi = 1$, implying that the volume flux of the fluid, and the velocity are equal [20]. However, when working with fluid flow in porous medium a distinction between these are needed.

3.5 Mass balance

The equation of continuity can be used to maintain conservation of mass in each phase of our system. For the gas phase the equation of continuity can be expressed as following [18]:

$$\varphi \frac{\partial \rho_g}{\partial t} + \nabla \cdot (\rho_g \vec{v}) = \varphi \dot{R} \tag{3.8}$$

Where ρ_g is the density of the gas phase. The equation of continuity describes the rate of change of the fluid density, at a fixed point in space. The first term describes the rate of increase per mass per unit volume. The seconds term describes the net rate of mass addition due to convection per unit volume, where $\vec{v_g}$ is a velocity vector denoting the superficial velocity of the gas. The source $\varphi \dot{R}$ is the mass added to the phase due to chemical reactions in the system, given in (kg/m^3). In Cartesian coordinated the equation of continuity can be expressed as:

$$\varphi \frac{\partial \rho_g}{\partial t} + \frac{\partial (\rho_g v_x)}{\partial x} + \frac{\partial (\rho_g v_y)}{\partial y} + \frac{\partial (\rho_g v_z)}{\partial z} = \varphi \dot{R} \tag{3.9}$$

It is assumed that the density of the solid and gas phase are constant. This implies that the chemical reactions have equimolar counter diffusion, however this does not apply to all the chemical reactions defined in this model. It is therefore assumed that the mass contribution can be neglected so that $\dot{R}$=0 in each phase. As we have one a one dimensional phase with z direction dependency and constant density, equation (3.9) can be simplified to:

$$\rho_g \frac{\partial v_z}{\partial z} = 0 \tag{3.10}$$

or

$$\frac{\partial v_z}{\partial z} = 0 \tag{3.11}$$

From the derived equations and simplifications made the solid and gas velocity in the system remain constant in the domain. From here on U_g and U_s denote the velocity of gas and solid respectively (with $U_{g,m}$ and $U_{s,m}$ denoting molar velocities) in the z-direction. As the densities and velocities in the system are constant, the total equation of continuity is not needed in order to calculate the velocities. However, in order to obtain conservation of species, species balances are needed for all species present in the system.

3.5.1 Conservation of species

As there are chemical reactions in the system, we want to consider the conservation of each chemical species, by using the mole based equation of continuity. Each chemical element is conserved, but this implies that ordinary chemical reactions involving the elements are merely changing partners, rather than being produced or or consumed. The individual species are not conserved as they can be generated or consumed in chemical reactions, but one can write an appropriate balance equation or transport equation for each of the chemical species involved in the chemical reactions [21]. Conservation of carbon was done using equation of continuity on mole basis as follows [22]:

$$(1-\phi)\frac{\partial C_C}{\partial t} + \frac{\partial U_{s,m} C_C}{\partial z} = (1-\phi)(-2R_1 - R_5), \tag{3.12}$$

where the reactions rates R_1 and R_5, both with units mol/m^{3}s, denote the degradation of carbon in the reactions 3.1 and 3.5. Note that the equation has been multiplied with the volume fraction of the solid phase, as this is the conservation of the species in their respective phases. Similar system equations have been used by Sloman et al. [5], but the porosity is baked in to the concentration.

The same process can be used to derive partial differential equations for the other chemical species, thus we obtain the following system equations for the solid and gas phase:

$$(1-\phi)\frac{\partial C_{SiC}}{\partial t} - \frac{\partial U_{s,m} C_{SiC}}{\partial z} = (1-\phi)(R_1), \tag{3.13}$$

$$(1-\phi)\frac{\partial C_{SiO_2(s)}}{\partial t} - \frac{\partial U_{s,m} C_{SiO_2(s)}}{\partial z} = (1-\phi)(-R_3), \tag{3.14}$$

$$(1-\phi)\frac{\partial C_{SiO_2(l)}}{\partial t} - \frac{\partial U_{s,m} C_{SiO_2(l)}}{\partial z}(1-\phi)(R_2 - R_{-2} + R_3), \tag{3.15}$$

$$(1-\phi)\frac{\partial C_{Si}}{\partial t} - \frac{\partial U_{s,m} C_{Si}}{\partial z} = (1-\phi)(R_2 - R_{-2}), \tag{3.16}$$

$$\phi\frac{\partial C_{CO}}{\partial t} - \frac{\partial U_{g,m} C_{CO}}{\partial z} = \phi R_1, \tag{3.17}$$

$$\phi\frac{\partial C_{SiO}}{\partial t} - \frac{\partial U_{g,m} C_{SiO}}{\partial z} = \phi(-R_1 - 2R_2 + 2R_{-2}). \tag{3.18}$$

In order to solve these equations the molar velocity of the solid and gas phase need to be calculated. This can be done through the relation that the mass flux, j_i, and molar flux, $j_{m,i}$ for species i:

$$\dot{j}_i = M_i j_{m,i} \tag{3.19}$$

$$U_{g,m} = \frac{\sum j_{m,i,g}}{C_g}, \quad i = SiO(g), CO(g) \tag{3.20}$$

$$U_{s,m} = \frac{\sum j_{m,i,s}}{C_s}, \quad i = C(s), SiC(s), SiO_2(l), SiO_2(s), Si(l) \tag{3.21}$$

where M_i is molar mass for species i, C_s and C_g denote the total concentration of solid-liquid species and gas species in their respective phases. $j_{m,i,s}$ and $j_{m,i,g}$ denote the total molar flux in the solid and liquid phase.

We will assume that the molar velocities are constant, this implies that we assume in-compressible fluid for the molar species, even though that is not the case for systems with chemical reactions. However, this was done to avoid build-up of species due to difference in molar velocities and mass velocities and to keep the system stable. In this model the radial dependency on concentrations is not taken into account. And the concentrations are therefore not radially dependant.

3.6 Velocity

The solid velocity has been assumed constant as it also was done by Sloman et al. [5]. How the solid velocity is calculated will be shown in section 5.1. The superficial gas velocity will be calculated using Darcy's law which can be expressed as [23, 18]:

$$U_g = -\frac{k}{\mu}\Delta p \tag{3.22}$$

Where U_g is the gas velocity along the z-axis, k (m^2) is the permeability of the charge, Δp ($\frac{\mathrm{kg}}{\mathrm{s}^2\mathrm{m}}$) is the pressure gradient due to pressure drop drop along the z-axis and μ ($\frac{\mathrm{kg}}{\mathrm{m}\cdot\mathrm{s}}$) is the dynamic viscosity of the gas mixture.

As we have chemical reactions in the system for simplicity it is assumed that both gasses, SiO(g) and CO(g), move with the same velocity U_g and solid and liquid components move at constant velocity U_s. The gas velocity is set to only vary with the pressure gradient, but as it has been assumed constant, the pressure gradient is constant in this framework.

3.6.1 Constant total pressure, varying pressure

In industrial furnaces the gas pressure has been observed to vary up to around 4 kPa above atmospheric pressure (101 kPa) [3]. These variations will be used to calculate the velocity of the gas phase according to Darcy's law, but will be kept constant. This means that we assumed an overpressure in the furnace hearth, $P_{hearth} = 20$ mbar. In order to calculate the partial pressures of the gaseous species, for simplicity we follow the approach used in the dynamic model of Harvorsen et al. [14], as Sloman et al [5] also did, keeping the total pressure constant. This constant, called P_{TOT} is set to 1 atm and is unrelated to the pressure used to calculate the gas velocity, and will only be used to calculate partial pressures of CO and SiO gas in order to calculate reaction rates.

3.6.2 Ideal gas mixture

The gas is modeled as being ideal, relating the total gas concentration, C_g, the total gas pressure, P_{TOT}, gas volume fraction for gas phase φ, gas constant, $R = 8.314$ J/molK, and temperature though the ideal gas law:

$$P_{TOT} = \frac{C_g R T}{\varphi} \tag{3.23}$$

The gas mixture is composed of CO and SiO gas, so the total concentration can be found as $C_g = C_{CO} + C_{SiO}$. Similarly,the total gas pressure is a function of CO and SiO partial pressures as follows:

$$P_{TOT} = P_{CO} + P_{SiO} \tag{3.24}$$

Dalton's law is outlined by Gaskel [24] from which we can deduce

$$\frac{P_{CO}}{P_{TOT}} = \frac{C_{CO}}{C_g} \tag{3.25}$$

and

$$\frac{P_{SiO}}{P_{TOT}} = \frac{C_{SiO}}{C_g}. \tag{3.26}$$

3.6.3 Reaction kinetics

The reaction rates R_j of the chosen reactions j are dependant on the local environment in the furnace such as partial pressures, chemical concentration and temperature [5]. From here on the reactions 3.1-3.5 will be denoted R_1-R_5. There are several factors that can influence the reaction rates, such as crystal structure, level of impurities, porosity, grains size of both carbon and quartz, these were discussed by Wiik [17]. Most parameters, such as constants in the reaction rate laws and constants determining rates of melting, are not known and difficult to measure due to high temperatures and SiO gas [15].

In this model the reaction rate functions presented by Sloman et al. [5] were used and are given in table 3.1. The rest of this section gives as short description of these equations and theory used to derive them. They utilized the ethos of Halvorsen et al. [15], utilizing simplified kinetic rates which sufficiently capture furnace behaviour. When they are determined by experimental procedures, these data can be included in the model developed in this work. Sloman et al. used ideas from relevant literature [15, 16, 17] among others. The reaction constants,k_i, were determined in section 5.12, but are defined such that the unit of the reaction rate is mol/m^{3}s.

Sloman et al. [5] followed Halvorsen et al. [15], where the kinetic laws are based on deviation from equilibrium. For each reaction an equilibrium pressure is defined as the driving force of the reaction for reactions involving SiO gas. Concentrations have been used to model solid and liquid reactants, while pressure difference from equilibrium in each reactions such as Halvorsen et al. [15], for the gas reactants. The function f^+ denotes the positive part of the function f. This means that $f^+ = \max\{f, 0\}$. Calculations of equilibrium partial pressures are presented in section 3.6.4. Arrhenius equation is commonly used to model the relationship between rate of a chemical reaction and temperature, using an activation energy, which acts as a barrier to the reaction. Lower activation energy gives a faster reaction. Arrhenius term $e^{-E_j/RT}$ is used, where E_j is the activation energy of reaction j, gas constant $R = 8.314$ J/molK and temperature T. Sloman et al. [5] sat the activation energies E_j to zero in order to simulate an industrial furnace. This was also adopted here. The reactivity of carbon, r_C, is treated as a constant which takes value between 0 and 1 [5]. In the stoichiometric model [1, 13] a similar approach was taken, a reactivity number described the fraction of carbon that will react in the upper part of the furnace. According to Sloman et al. [5] this approach allows for observation of the influence of more or less reactive materials on furnace dynamics.

Table 3.1: Reaction rates used for each reaction taken from Sloman et al. [5]

Reaction	Reaction Rate
R_1	$k_1 r_c C_C \left(P_{SiO} - P_{SiO,equ,1}(T_g)\right)^+$
R_2	$k_2 e^{\frac{-E_2}{RT}} \left(P_{SiO} - P_{SiO,equ,2}(T_g)\right)^+$
R_{-2}	$k_{-2} e^{\frac{-E_{-2}}{RT}} C_{SiO_2(l)} C_{Si}$
R_3	$k_3 e^{\frac{-E_3}{RT}} C_{SiO_2(s)} \left(T_s - T_m\right)^+$
R_4	$k_4 e^{\frac{-E_4}{RT}} C_{SiC(s)} \left(P_{SiO} - P_{SiO,equ,4}(T_g)\right)^+$
R_5	$k_3 e^{\frac{-E_3}{RT}} C_{SiO_2(s)} \left(T_s - T_m\right)^+$

3.6.4 Equilibrium pressure functions

Expressions derived by Sloman et al. [5] are used to calculate the equilibrium partial pressure functions used to describe the reaction rate for reactions R_1, R_2 and R_4. Here the Gibbs free energy of a reaction, ΔG_{Rj} (J/mol) and the equilibrium constant, K_{equ} are related through the reaction isotherm [24]:

$$\ln(K_{equ}) = \frac{-\Delta G_R}{RT} \tag{3.27}$$

The chemical reaction constant can be expressed by the ratio of products k_+ to reactants k_- as $K = \frac{k_+}{k_-}$. It is assumed that the activity of the gases is given by their partial pressure. We also assume unitary activity for the solid and liquid species in the reactions. Then the relations $P_{CO} = P_{TOT} - P_{SiO}$ was utilizes and yielded following expressions:

$$P_{SiO,equ,j} = \frac{P_{TOT}}{1 + \exp\left(\frac{-\Delta G_{R_j}}{RT}\right)}, \quad j = 1, 4, \tag{3.28}$$

$$P_{SiO,equ,2} = P_{TOT} \min\left\{1, \exp\left(\frac{\Delta G_{R_2}}{2RT}\right)\right\} \tag{3.29}$$

Here j denoted reaction rate R_j or reaction j. According to Gaskel [24], the simplest way to find ΔG_{R_j} for each reaction is to fit the data for the constants a_j, b_j and d_j in equation 3.30. This was done for a heat and mass transfer model developed by Sloman et al. The same values presented are used here and listed in table 3.2.

$$\Delta G_{Rj}(T) = a_j + b_j T ln(T) + d_j T. \tag{3.30}$$

Table 3.2: This table gives an overview of the parameter values used for calculations of the equilibrium partial pressure functions. These are taken from Sloman et al. [5].

j	a_j **[J/mol]**	b_j **[J/molK]**	d_j **[J/molK]**
1	-8.183 $\times$ 10^4	-1.687	1.528 $\times$ 10
2	-8.178 $\times$ 10^5	-9.903 $\times$ 10	1.141 $\times$ 10^3
4	-9.901 $\times$ 10^4	-1.264 $\times$ 10^2	1.109$\times$$10^3$

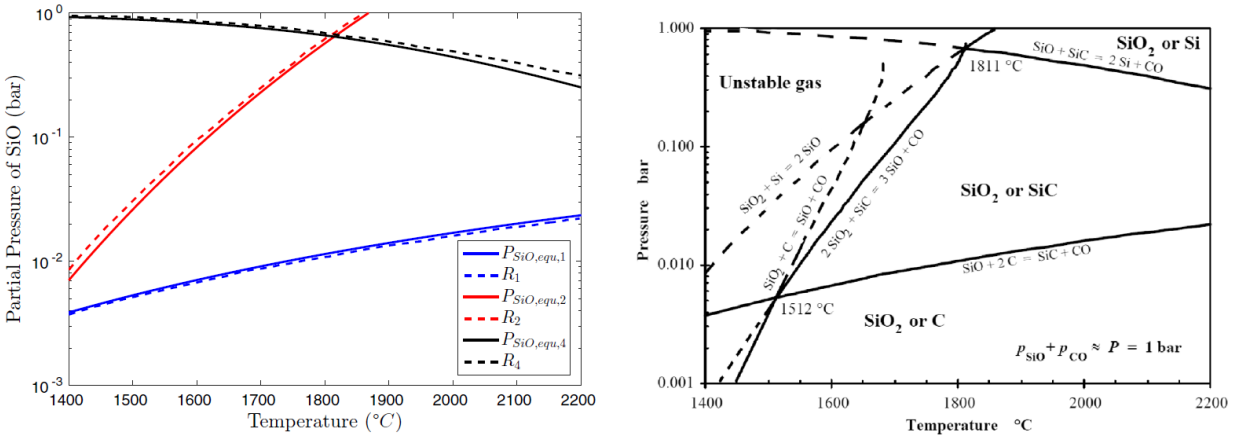


Figure 3.3: To the left equilibrium pressure functions, solid lines, are compared with read in data from the image on the right, dashed lines, taken from Sloman er al. [5]. To the right is the diagram of equilibrium SiO pressures taken from Schei et al. [1].

3.7 Conservation of Energy

In this sub-chapter equations for conservation of energy in the system will be derived. The solid phase is considered a porous medium in which the gas flows through. The solid porous medium is assumed to be isotropic, and viscous dissipation, radiative effects and work done by pressure changes are negligible. It is assumed that we have no heat generation due to electrode heat, this implies that we are far enough away from the electrode that there is no heat contribution from the electrode. As we have defined two phases in our system, two temperature equations are needed, the derivation of them are based on the book *Convection in Porous Media* by Nield and Bejan [18]. However, Nield and Bejan assume that the porous material is in a fixed position, while in our system the solid porous material (the raw materials) move downwards, while the fluid (the gas) flows upwards. In our model it is assumed that there is heat transfer between the solid (joined solid and liquid) and gas phase, and therefore absence of local equilibrium. Using theory presented by Nield and Bejan [18] the energy balance for the solid phase can be expressed by taking averages over an elemental volume of the medium, adding the assumption that the solid is in movement by adding a convective term in such a manner:

$$(1-\varphi)(\rho c)_s \frac{\partial T_s}{\partial t} + (\rho c)\vec{v}_s \cdot \nabla T_s = (1-\varphi)\nabla \cdot (k_s \nabla T_s) + (1-\varphi)q_s''' + h(T_g - T_s). \tag{3.31}$$

The energy balance of the gas phase can be expressed as following [18]:

$$\varphi(\rho c_p)_g \frac{\partial T_g}{\partial t} + (\rho c_p)\vec{v}_g \cdot \nabla T_g = \varphi\nabla(k_g \nabla T_g) + (\varphi)q_g''' + h(T_s - T_g). \tag{3.32}$$

The subscripts s and g refers to solid and gas phases respectively, $\vec{v}_s$ is the velocity of the solid phase, an $\vec{v}_g$ is the superficial velocity of the gas phase. c is the specific heat of the solid, c_p is the specific heat at constant pressure of the fluid, k_s and k_g is the thermal conductivity of solid and gas phase respectively, and $q_s^{'''}$ and $q_g^{'''}$ are the heat production per unit volume with units W/m^3. The heat transfer coefficient is denoted as h with units W/m^{3}K. In the above equations it is assumed that surface porosity is equal to the porosity as this is very relevant considering the conductive terms in the energy balance equations. In equation (3.31) the term $\nabla \cdot (k_s \nabla T_s)$ expresses the net rate of heat conduction into a unit volume of the solid. The terms are multiplied with $(1-\varphi)$ because this is the ratio of volume occupied by solid to the total volume of the element.

3.7.1 Simplification of the energy balance

For simplicity, thermal conduction is neglected in order to develop a simple framework upon which further equations can be added. We assume that only the reactions are a source of heat production in the system. Equations (3.31) and (3.32) can be simplified to:

$$(1-\varphi)(\rho c)_s \frac{\partial T_s}{\partial t} + U_s(\rho c)_s \frac{\partial T_s}{\partial z} = (1-\varphi)q_s^{'''} + h(T_g - T_s) \tag{3.33}$$

$$\varphi(\rho c_p)_g \frac{\partial T_g}{\partial t} + U_g(\rho c_p)_g \frac{\partial T_g}{\partial z} = \varphi q_g^{'''} - h(T_g - T_s) \tag{3.34}$$

Considering that the model is one dimensional, the system is in Cartesian coordinates and only allows movement along the z-direction, both the equations above can be simplified to:

$$(1-\varphi)(\rho c)_s \frac{\partial T_s}{\partial t} + U_s(\rho c)_s \frac{\partial T_s}{\partial z} = (1-\varphi)q_s^{'''} + h(T_g - T_s) \tag{3.35}$$

$$\varphi(\rho c_p)_g \frac{\partial T_g}{\partial t} + U_{z,g}(\rho c_p)_g \frac{\partial T_g}{\partial z} = \varphi q_g^{'''} - h(T_g - T_s) \tag{3.36}$$

The heat production or consumption due to the chemical reactions for one phase can be expressed as:

$$q_s^{'''} = -\sum_i R_i \Delta H_{i,s} = -(R_1 \Delta H_{1,s} + R_2 \Delta H_{2,s} + R_{-2} \Delta H_{-2,s} + R_3 \Delta H_{3,s}) \tag{3.37}$$

Where R_j is the reaction rate for reaction j, and $\Delta H_{j,s}$ is the reaction heat contribution to the solid phase due to reaction j. How these were calculated will be further explained in section 5.11. It should also be noted that even though the gas velocity is set to be the superficial velocity $U_{z,g}$, the the temperature moves with the average fluid velocity $v_{z,g}^{'} = \frac{U_{z,g}}{\varphi}$. This can be shown by assuming no source terms as follows by using the energy balance for the gas phase:

$$\varphi(\rho c_p)_g \frac{\partial T_g}{\partial t} + U_{z,g}(\rho c_p)_g \frac{\partial T_g}{\partial z} = 0 \tag{3.38}$$

Dividing by $(\rho c_p)_g$ yields:

$$\varphi \frac{\partial T_g}{\partial t} + U_g \frac{\partial T_g}{\partial z} = 0 \tag{3.39}$$

Using $v_{z,g}^{'} = \frac{U_{z,g}}{\varphi}$, we get

$$\frac{\partial T_g}{\partial t} + v_{z,g}^{'} \frac{\partial T_g}{\partial z} = 0 \tag{3.40}$$

As we wish to allow for heat transfer between the two phases, determining the value of the heat transfer coefficient is critical according to Nield and Bejan [18]. According to correlations for a porous bed of particle established in Dixon and Cresswel [25], the heat transfer coefficient, h, can be expressed as:

$$h = a_{gs} h^* \tag{3.41}$$

where a_{sg} is the specific surface area(surface per unit volume) which is given by:

$$a_{gs} = \frac{6(1-\varphi)}{d_p} \tag{3.42}$$

And

$$\frac{1}{h^*} = \frac{d_p}{Nu_{gs} k_g} + \frac{d_p}{\beta k_s}, \tag{3.43}$$

where d_p is the particle diameter. Assuming that the particles have a spherical form and therefore the simplification $\beta = 10$ can be made [18]. The Reynolds number Re_p is based on particle diameter. If the Reynolds number, Re_p, has a value above 100, it correlates well with the gas-to-solid Nusselt number, Nu_{gs}, correlated by the expression [18]:

$$Nu_{gs} = \frac{h d_p}{k} = (\frac{0.255}{\varphi}) Pr^{\frac{1}{3}} Re_p^{\frac{2}{3}} \tag{3.44}$$

According to Nield and Bejan [18], for low values of Re_p the estimate of Nu_{gs} vary between 0.1 and 12.4 these being based on Miyauchi et al. [26] and Wakao et al. [27, 28]. Other authors have used alternative expressions for h^* and a_{gs}, which can be found in Nield and Bejan [18].

3.8 High Temperature Model

The high temperature model is based on the low temperature model with some changes. The domain of the model in the high temperature zone of the furnace, with a height of 1 m, in accordance with [1, 3]. The mesh consists of 100 quadratic cells. We assume that there is gas flowing in from the bottom of the model with gas composition of 50/50 SiO and CO partial pressures and temperature of 2000 °C. We also assume that there is heat generation due to the electrode contribution to the heat balance equation.

3.8.1 Electrode heat generation

We assume that the electrode heat generation is mainly at height $z = 0$, and that we can express the heat generation due to electrode heat generation from the electrode using an exponential function:

$$Q_{el} = Q_{el,a} * (e^{Q_{el,b} * (1 - z/L)} - 1) \tag{3.45}$$

Q_{el} (W/m^3) is the energy added per volume. $Q_{el,a}$ (W/m^3) and $Q_{el,b}$ (-) are constants calculated in section 5.8.

3.8.2 Energy balance

For the high temperature model, Q_{el} has been added as a source term in the temperature equations as following:

$$(1-\varphi)(\rho c)_s \frac{\partial T_s}{\partial t} + v_{z,s}(\rho c)_s \frac{\partial T_s}{\partial z} = (1-\varphi) q_s''' + h(T_g - T_s) + 0.1 Q_{el} \tag{3.46}$$

$$\varphi(\rho c_p)_g \frac{\partial T_g}{\partial t} + v_{z,g}(\rho c_p)_g \frac{\partial T_g}{\partial z} = \varphi q_g''' - h(T_g - T_s) + 0.9 Q_{el} \tag{3.47}$$

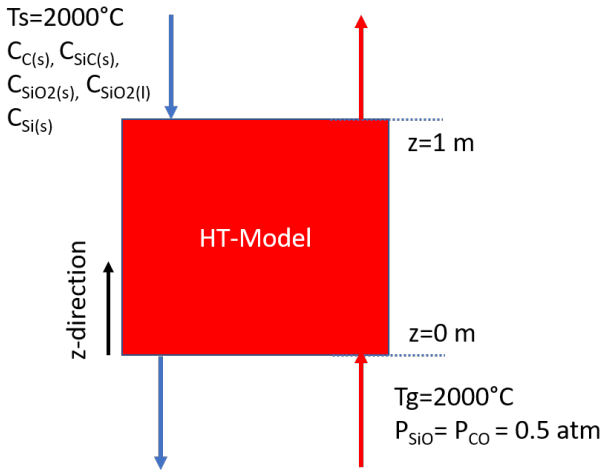


Figure 3.4: The principle flow chart of the high temperature model created in this thesis.

3.8.3 Velocity

The mass velocities of the solid and gas phase are taken from calculations for the low temperature zone, and also assumed to be constant. The molar velocities are also based on the calculated values for the LT-model.

Chapter 4

Computational Fluid Dynamics and OpenFOAM

This chapter gives a description of some important concepts in Computational Fluid Dynamics (CFD) and OpenFOAM used in this work. The equations described in the previous chapter are a set of nonlinear partial differential equations. These rarely have an analytical solution. In order to give detailed description of the practical problems, Computational Fluid Dynamics applies numerical methods. An overview of some concepts of CFD are given in this section, with focus on subjects relevant for this thesis. A more detailed introduction to CFD can be found in Versteeg and Malalasekera [29] which will be used here.

4.1 Mesh

A computer is discrete by nature, therefore values must be defined in points instead of being a continuous value. The result is a set of values at certain points. Interpolation is used to describe the values between these points. By creating a mesh, the user can set the locations and amount of points. In figure 4.1 an illustration of a simple mesh in two dimensions is presented. In figure 4.2, a sketch of one individual cell is shown. For each cell in the mesh, conservation laws and momentum balances will be solved. Each cell have faces, these define the borders between them. Boundary conditions are enforced by defining the values of the outer face of points on the edges of the system.

Finer mesh yields more accurate solutions, however, momentum balance and conservation laws must then be solved at even more locations. This results in higher computational time due to increase amount of equations that need to be solved.

4.2 Finite Volume Method

The finite volume method is a commonly used method in CFD, in which the computational domain is divided into a mesh of *control volumes*, for which conservation equations are solved on integral form. An example used by Versteeg and Malalasekera [29] is the mass conservation for an in-compressible fluid in steady-state $\nabla \cdot \vec{u} = 0$, which can be written as:

$$\int_{C.V} \nabla \cdot \vec{u} dV = \int_{C.S} \vec{u} \cdot \vec{n} dA = 0 \tag{4.1}$$

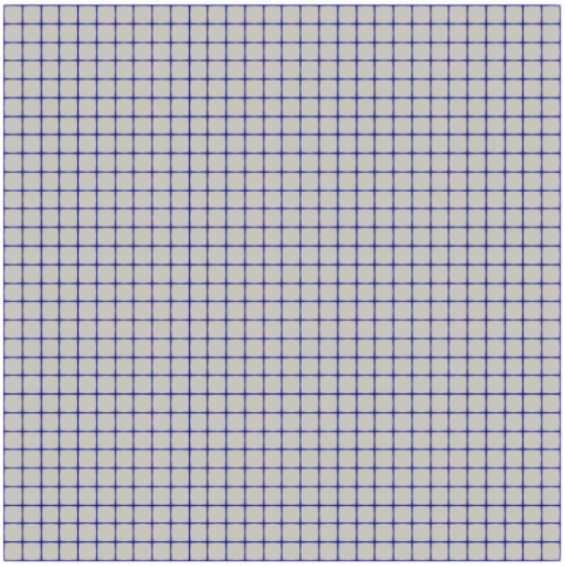


Figure 4.1: Illustration of a mesh in two dimensions, in total 900 cells. Illustration is taken from [30]

In equation (4.1) the the divergence theorem was applied. Considering a 2D volume as sketched below, the velocity at face f can be written in vector notation:

$$\vec{u_f} = [u_f, v_f] \tag{4.2}$$

And hence by using the same cells as shown in figure 4.2 Equation. 4.1 can be written as:

$$-u_e\Delta y - v_s\Delta x + u_w\Delta y + v_n\Delta x = 0 \tag{4.3}$$

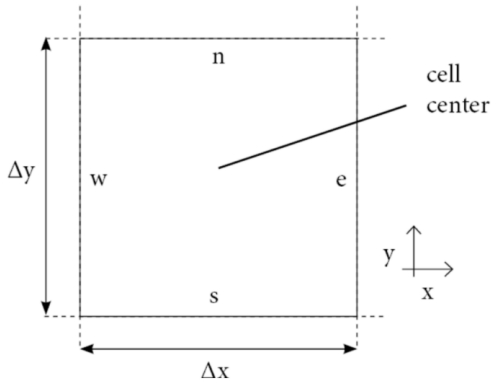


Figure 4.2: Sketch of a 2D cell. The cell has four faces and velocities on each face. Taken from [30]

Equation (4.3) is a discrete equation which describes how the velocities on the face of the volume are connected. The single equation describes a single cell. The system consist of several cells, where each cell in the problem will have a corresponding equation. As each internal face is shared between two cells, it yields a coupled set of algebraic equations, ultimately leading to a matrix equation of the form

$$Ax + B = y \tag{4.4}$$

where A is the system matrix, x is the matrix of unknown variables and y is a vector containing values independent of where the unknowns is located. In order to solve for x, A needs to be inverted. However, a direct inversion of a matrix if often hard to apply as in normal cases, the matrix will consist of tens of thousands or even millions entries. In these cases iterative methods can be used. Discussion of iterative methods are beyond the scope of this project, but detailed descriptions and algorithms can be found in Versteeg and Malalasekera [29].

4.3 Convergence

CFD solves the problems by iterations. These will stop when convergence is reached. At convergence, the values do not change after additional iterations. A slightly looser convergence tolerance is used, as iterating until none of the values change will take an very long time. Discrete equations are applied to inner points, while a combination of discrete equations and boundary conditions are used on the outer points in the general situation. This results in a system with N independent variables and N x N matrix. As the number of grid points increase, the matrix will become very large, and correspondingly resource intensive. However, increasing the number of grid points reduces Δx and gives better resolution, increases the accuracy of the calculations as the discretization error $O(\Delta x)$ decreases. As the solution approaches the analytical solution we say that it is "grid converged". This is typically determined by some user defined tolerance (within 1 $\%$ of analytical solution). Only grid converged solutions should be trusted - or else the error might be arbitrary high.

$$(\frac{du_i}{dx})_i = \frac{u_i - u_{i-1}}{\Delta x} + O(\Delta x) \tag{4.5}$$

It should be noted that iterative convergence and grid convergence measure two different errors: the iterative error is $O(\Delta u_i)^2$ and is due to approximations of the non-linear terms. The discretization error on the other hand arises from the approximation of the gradient. The discretization error becomes smaller the greater number of grid points, while the iterative error decreases the greater number of iterations.

4.4 Stability

Time dependent problems must be discretisized in both time Δt and space Δx. The time step is limited by the spatial resolution in the sense that too large time steps will give unstable solutions which typically result in the simulation crashing. A time step that is too large is dependant on the algorithm, but a practical rule is to use the Courant number

$$Co = \frac{\tilde{u} \cdot \Delta t}{\Delta x} < 1 \tag{4.6}$$

Here Co is the courant-number and $\tilde{u}$ is a typical velocity. The Courant-Fredrich-Lewy relation gives the maximum time step

$$\Delta t_{max} = \frac{\Delta x}{\tilde{u}} \tag{4.7}$$

4.5 Vertification and validation

An error is defined in the literature as a recognisable deficiency in a CFD model that is **not** caused by lack of knowledge. Uncertainties, however, are potential deficiencies in a CFD model that are caused by the lack of knowledge [29]. Numerical errors, coding errors and user errors are typical causes of errors. Input uncertainties such as limited knowledge of geometry, boundary conditions, properties or physical model uncertainties (such as difference between real flows and CFD due to insufficient representation of physical properties) are typical causes of uncertainties. A verification of the model quantifies the errors, this is a process determining if the implementation of the model accurately represents the conceptual description of the problem. Verification of a model is traditionally performed by evaluating global and local conservation equations in addition to comparing simplified parts of the model to analytical solutions. Validation on the other hand is traditionally carried out by comparison with experimental data. Comparing key features that are also obtainable from the CFD model.

4.6 OpenFOAM

The framework developed in this work is implemented in OpenFOAM. This section contains details about the setup of simulations, and descriptions of the developed solver created during this work. OpenFOAM is a free, open source simulation platform [31]. The platform has a wide range of applications, and can be used to solve complex fluid flows involving chemical reactions, heat transfer, and turbulence. A drawback is the varying level of documentation. C++ is used to write files and user can write solvers from scratch. When a simulation is completed the data must be visualized in another program, such as ParaView. There are several inbuilt solvers in in OpenFOAM. Two solvers were developed from laplacianFOAM in this work [32]. We will only show the solver for the low temperature zone in this section, the difference will be shown in the appendix A as the two solvers have many similarities.

The solver contains the source code for the model, for example how to calculate velocity, heat transfer, species balance and energy balance calculations. The functionalities and file developed in this solver are presented in table 4.1. The solver also includes files about how the setup should be, for example numerical methods, mesh, and initialization of all constants and fields in the framework. A flow sheet of how this works is presented in figure 4.3. Overview and description of the different dictionaries are shown in table 4.2. Constants are defined in a folder called Constants, an overview is presented in table 4.3. In the 0 file all concentrations, velocities, pressure profiles, temperature profiles are initialized, and boundary conditions are also set here. One of the main advantages of OpenFOAM is that for each specifies writeInterwall created a new folder for all the fields initialized in the 0-folder these are the results that can be vizualized. Another advantage is that the boundary conditions can be changed, when entering field in the last created time folder, then changing the boundary condition there. Then the simulation can continue with a new bondary conditon. This can (for instance) be done to change the concentration of carbon entering the low temperature zone, after the simulation has run in order to study the dynamic affects this has on the furnace.

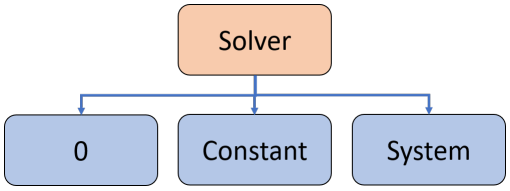


Figure 4.3: Illustration of different directories in ergunMasterFOAM.

Table 4.1: Overview of the different files found in the solver that were implemented in this work

Name	Purpose
ergunMasterFoam.C	Main source file, calculations of gas velocity
createfields.H	Declaration of all variabeles used in the framework
heatExchangeCoeffCalc.H	Calculation of the heat transfer coefficient
massBalance	Conservation equations for each species are implemented here
pressureCalc.H	Pressure calculations
reactionHeat.H	Calculates reaction enthalpy, Reaction heat contribution, specific heat capacity to the solid and gas phase.
viscocity.H	Calculation of viscosity of the gas phase
reactionRate.H	Calculation of reaction rates and equilibrium pressure functions
themCond.H	Implementation of calculation of thermal conductivity
elHeatSource	Calculations of energy contribution due to electrode heat generation

Table 4.2: Overvies of files in system directory

Name	Purpose
blockMeshDict	Defines how the mesh of the system should be created
fvSolution	Stated the numerical methods that should be used and iterations algorithm
controlDict	States the writing interval, time interval and time step
fvSchemes	States the discretization schemes that should be used

Table 4.3: Overvies of files in the constant directory

Name	Purpose
chemProperties	Molar mass of each chemical species
enthalpyCalc	Constants used to calculate enthalpy and heat capacity
dynamicViscosity	Constants used to calculate viscosity of the gas phase
themDat	Activation energy for each reaction, constants used to calculate equilibrium partial pressures, gas constant, and melting temperature of quartz
rateConstants	Rate constants for each reaction
thermalCondCoeff	Constant used to calculate thermal conduction
transportProperties	Properties describing the constant used to calculate gas velocity

Chapter 5

Parameter Determination

In this chapter the determination of the system parameters used in the framework are derived. The framework presented in chapter 3 has several equations describing different aspects of the LT and HT model. In order to allow for more representative parameters for the silicon process, several of the parameters are calculated based on a fictional furnace using the stoichiometric Elkem model with a furnace load of 30000 kW. The data are presented by Tveit [19] as several parameters are calculated based on furnace dimensions, rate of raw material consumption and gas production. The data from the fictional furnace are presented in table 5.1.

The main purpose of this thesis is not hindered by the lack of data available to calculate the system parameters as the aim of this work is to demonstrate that the framework can capture dynamic changes inside the domain and given reasonable results with system parameters of right order of magnitude. When more accurate values are found and implemented in the model, it should return similar, but more accurate results. All final results are presented in the end of each subsection in either a table or specified in the text in SI-Units as implementation in OpenFOAM requires it.

Table 5.1: Furnace dimensions, rate of raw material consumption, gas production and porosity of charge used for determination of several parameters in this work. Data are obtained by private communication with Tveit [19] and are based on calculations on a fictional furnace using the stoichiometric Elkem model with a furnace load of 30 000 kW. The table was also used in the project thesis of Hajdini [33]

Parameter	**Value**	**Units**
Amount of dry quartz, $\dot{m}_{SiO_2}$	7000	kg/h
Amount of added coal, $\dot{m}_C$	4600	kg/h
Bulk density of charge, ρ_{bulk}	1500	kg/m^3
Internal diameter of furnace, $d_{internal}$	9.5	m
Height of charge, H_{charge}	2,5	m
Electrode diameter, d_{el}	1550	mm
Number of electrodes, N_e	3	-
Temperature of gas mixture, T_g	1900	°C
Cavity, φ	30	%
Rate of SiO(g) production, $\dot{m}_{SiO}$	946	kg/h
Rate of CO(g) production, $\dot{m}_{CO}$	5276	kg/h

5.1 Solid velocity

The solid velocity is an important parameter to determine in this thesis and determines how fast the solid phase moves downwards in our framework. The area in which the solid moves through is calculated based on the assumption that around 70 % of the charge is consumed in a radius 1 meter away from each electrode [19]. According to Tveit [19] it is not possible to observe the movement of the solid charge on the top of the furnace with the naked eye. The parameter U_s (m/s) should therefore expected to have a small value. In order to calculate a solid velocity data from the fictional furnace presented in table 5.1 will be used. The total mass flow rate of the charge can be calculated as in equation 5.1 where the total mass flow of solid charge is represented by $\dot{m}_s$, ρ_{bulk} is the density of the bulk solid phase and $A_{ac,tot}$ is the total active area around the three electrodes.

$$0.7\dot{m}_s = \rho_{bulk}\dot{V} = \rho_{bulk}A_{ac,tot}U_s, \tag{5.1}$$

Rearranging the above equation and dividing with 3600 s/h to achieve SI-units yields an expression for the solid velocity.

$$U_s = \frac{0.7\dot{m}_s}{3600\rho_{bulk}A_{ac,tot}}, \tag{5.2}$$

The area around the three electrodes was calculated by using the inner diameter of the electrode, d_{el}, values were taken from table 5.1, with active radius around each electrode set to $R_{ac} = 1$ m. The active area as was calculated as in the following equation firstly calculating the active area around each electrode, $A_{ac,el}$, then multiplying this with the number of electrodes in the furnace:

$$A_{ac,el} = \pi\Big(\frac{d_{el}}{2} + R_{ac}\Big)^2 - \pi\Big(\frac{d_{el}}{2}\Big)^2 \tag{5.3}$$

$$A_{ac,tot} = 3 \cdot A_{ac,el} \tag{5.4}$$

The total mass flow of the solid phase is the sum of the mass flow of added charge to the furnace:

$$\dot{m}_s = \dot{m}_{SiO_2} + \dot{m}_C \tag{5.5}$$

The values of the mass flow of the solid phase are taken from table 5.1. The calculation yielded a solid velocity of 6.25E-05 m/s.

5.2 Gas velocity

The gas velocity is calculated in OpenFOAM using Darcy's law [22, 3] which can be expressed using the permeability of the charge, k (m^2), the pressure drop, Δp (kg/m·s^2), along the z-axis and the dynamic viscosity of the gas mixture, μ (kg/m·s), as:

$$U_g = -\frac{k}{\mu}\Delta p \tag{5.6}$$

According to Kadkhodabeigi [3] the crater pressure increase is typically 20-30 mbar depending on furnace operation. From industrial measurements with furnace load of 30 MW, the pressure increase is about 20 mbar. The pressure gradient in the model is calculated in OpenFOAM by initializing the pressure at the inlet and outlet of the LT. It is assumed that the pressure at the top is 1 atm as we assume that the furnace is open.

Permeability is a measure of ease of passage of gases or liquids through a porous material [34]. The permeability used in this thesis was determined by Hajdini [33] by calculating the gas velocity in communication with Tveit [19], using [35] and rate of gas production presented in table 5.1. The permeability was then fitted in OpenFOAM so that is corresponded with the calculated gas velocity with Tveit [19].

Average dynamic viscosity for the temperature interval was calculated in the project thesis Hajdini [33], which is used here as it has been assumed that the velocity is constant.

The calculations yielded: $P_{inlet} = 103325$ (kg/m·s^2) and $P_{outlet} = 101325$ (kg/m·s^2), $k = 7.815 \cdot 10^{-5}$ and $U_g = 1.29$ (m/s).In the HT-model it is assumed that the gas velocity is constant, and is calculated based on results from the LT-model. It should be noted that this is the superficial velocity, not the intrinsic velocity as presented in the theory.

5.3 Molar velocities

The molar velocities are calculated using the total molar flux, $\dot{j}_{tot}$ and C the total molar concentration initially in each phase as follows. The calculation for the gas phase will be shown here, as calculations for the solid phase is similar.

$$U_{m,g} = \frac{\dot{j}_{m,g}}{C_g} = \frac{\sum \dot{j}_{m,i,g}}{C_g} \tag{5.7}$$

It is possible to calculate the molar velocity from the mass velocity using the relationship:

$$\dot{j}_{m,i,g} = \frac{\dot{j}_{i,g}}{M_i} \tag{5.8}$$

Where $\dot{j}_i$ is the mass flux of species i, M_i (kg/mol) is the molar mass of species i. The mass fluxes for the species present in the gas phase were calculated by using the weight fraction, W_i (-), of the species, density of the gas phase and intrinsic velocity of the gas phase.

$$\dot{j}_{i,g} = W_{i,g} \cdot \rho_g \cdot U_g \cdot \frac{1}{\varphi} \tag{5.9}$$

An error was done here by dividing the superficial velocity U_g with porosity, as the molar velocities used also are superficial. The error due to this will be that the molar velocities are higher than what they are supposed to. Due to lack of time, this was not corrected in this work, and should be in further work. The weight fraction of each species was calculated using the concentration of each species in the gas phase to calculate a mass concentration (density) and a total mass concentration as follows:

$$W_{i,g} = \frac{\rho_{i,g}}{\rho_{tot,g}} = \frac{C_{i,g}/Mm_i}{\sum C_{i,g}/Mm_i} \tag{5.10}$$

Here $\rho_{i,g}$ is the mass concentration of species i in the gas phase. and the total mass concentration of the gas is $\rho_{tot,g}$ (kg/m^3). Even though it has been assumed that the density is constant, there are chemical reactions in the model, this was therefore done to make sure that the sum of the mass fractions would be 1.

5.4 Species concentration

In this thesis we are interested in the concentrations of species present in the active area in the furnace. In the theory it was assumed that the solid and liquid species are only present in the solid phase, while the gas species are only present in the gas phase. The concentrations are defined in each phase so that for species i in the solid phase:

$$C_{s,i} = \frac{n_{s,i}}{V_s} \tag{5.11}$$

and for gas species:

$$C_{g,i} = \frac{n_{g,i}}{V_g} \tag{5.12}$$

Here the V_s and V_g are the volume of the solid and gas phase respectively in the active area of the furnace. These volumes are calculated using the porosity of the charge and furnace dimensions presented in table 5.1 as follows:

$$V_s = V_{ac}(1 - \varphi) \tag{5.13}$$

$$V_g = V_{ac}\varphi, \tag{5.14}$$

where the volume of the active area, V_{ac}, is calculated using the height of the charge, H_{charge}, and active area around the three electrodes:

$$V_{ac} = H_{charge}A_{ac,tot} \tag{5.15}$$

In order to calculate the initial concentrations in the furnace, it is assumed that only quartz and carbon are present in the solid phase, and only silicon-monoxide and carbon-monoxide in the solid phase. We will assume that the furnace is empty and then filled with quartz and carbon in addition to the gas with their respective velocities. The total moles of the species in the solid phase in the active area of the furnace can be calculated using the total mass flow of each species presented in table 5.1, the time it takes to fill an empty furnace with solid with the calculated solid velocity

$$n_{s,i} = \frac{m_{s,i}}{M_i} = 0.7\frac{\dot{m}_{s,i}}{3600}\frac{1}{M_i}t_s = 0.7\frac{\dot{m}_{s,i}H_{charge}}{3600U_s}\frac{1}{M_i} \tag{5.16}$$

The rate is divided by 3600 in order to get SI-units. The initial concentration of SiO(g) and CO(g) was calculated using HSC, reaction SiO + SiC=Si + CO(g) + SiO(g) as the reaction equation and the relation of molar volume at different temperatures though ideal gas law:

$$V_m(T = 2273\text{K}) = \frac{2273\text{K}}{298\text{K}}V_m(T = 298\text{K}) \tag{5.17}$$

Which yielded the initial concentration and boundary concentrations for the species present in the furnace. These are presented in table 5.2.

Table 5.2: Initial and boundary concentration of each species in the furnace. $C_{initial,i}$ represents the initial bulk concentration of species i, and $C_{b,i}$ represents the boundary concentration of species i

Species	$C_{i,initial}$ $\frac{mol}{m^3}$	$C_{b,i}$ $\frac{mol}{m^3}$
SiO(g)	5.849	5.849
CO(g)	5.849	5.849
SiO_2(s)	2.152E4	2.152E4
C(s)	7.075E4	7.075E4

5.5 Choice of particle diameter

According to Schei et al. [1], the average size of quartz varies between 10-150 mm, while the particle size for carbon varies between 1.5-30 mm. The average particle diameter was therefore calculated using the largest diameters of carbon and quartz. The two diameters were weighted using the weight fraction, W_i, of the solid raw materials presented in table 5.1 to calculate the average particle diameter d_p as following:

$$d_p = W_{s,\mathrm{C}} d_{p,\max \mathrm{C}} + W_{s,\mathrm{SiO_2}} d_{p,\max \mathrm{SiO_2}} \tag{5.18}$$

Ideally it would be better to find the particle size distribution in order to calculate the average particle size, but I was not able to find that during this work. The calculations yielded a particle diameter $d_p = 80$ mm.

5.6 Density of solid and gas phase

The density of the solid and gas phases are constant in this framework. The density of the gas phase (which consists of a mixture of CO and SiO gas) is 0.23 kg/m^3 in furnace operating temperature according to Kadkhodabeigi [3]. The density of the solid phase is calculated based on the density of the raw materials carbon and quartz added to the furnace. The solid density is weighted by the solid mass fractions according to equation 5.19.

$$\rho_s = W_{SiO_2} \rho_{SiO_2} + W_C \rho_C \tag{5.19}$$

Here the mass fractions are calculated using the mass flow rate of solid carbon and quartz added to the furnace using the values given in table 5.1.

$$W_i = \frac{\dot{m}_i}{\dot{m}_s} \tag{5.20}$$

The densities of C(s) and SiO_2(s) are taken from SI Chemical Data [36] at 25 °C, as it has been assumed in the theory that the density of the solid phase remains constant in the temperature interval. The density of carbon was set to 2.3 kg/m^3, while the density of quartz was 2.6E3 kg/m^3. The density of the solid phase was calculated to b 2.481E3 kg/m^3.

5.7 The heat transfer coefficient

In order to calculate the heat transfer coefficient, a temperature dependent function for the kinematic viscosity was implemented. However, during simulations it was noticed that the heat transfer coefficient calculated was too high, as it yielded odd temperature profiles. In communication with Halvard Tveit, the heat transfer coefficient was therefore set to 200 (W/m^3).

Nevertheless, a temperature dependent heat transfer coefficient was implemented in the framework, and is therefore presented here. Excel was used to calculate a temperature dependant kinematic viscosity based on data of the kinematic viscosity of CO (g) taken from [37] presented in table 5.4. The calculations yielded the function presented in table 5.3

Table 5.3: Temperature dependent kinematic viscosity function implemented as $\nu = \nu_1 T_g^4 + \nu_2 T_g^3 + \nu_3 T_g^2 + \nu_4 T_g + \nu_5$, calculated using [38]

Function	$\nu_1 \frac{m^2}{sK^4}$	$\nu_2 \frac{m^2}{sK^3}$	$\nu_3 \frac{m^2}{sK^2}$	$\nu_4 \frac{m^2}{sK^1}$	$\nu_5 \frac{m^2}{s}$
$\nu(T)$	4.507E-17	-1.378E-13	2.067E-10	7.980E-10	-3.632E-08

Table 5.4: Kinematic viscosity of CO (g) at 1 atm pressure used in calculations of kinematic viscosity taken from [38]

Temperature T, ° C	**Kinematic viscosity** ν, m^2/s
0	$1.303 \cdot 10^{-5}$
100	$2.274 \cdot 10^{-5}$
150	$2.830 \cdot 10^{-5}$
200	$3.426 \cdot 10^{-5}$
300	$4.722 \cdot 10^{-5}$
400	$6.136 \cdot 10^{-5}$
500	$7.653 \cdot 10^{-5}$
1000	$1.700 \cdot 10^{-4}$
1500	$3.284 \cdot 10^{-4}$
2000	$6.543 \cdot 10^{-4}$

5.8 Electrode heat generation

The electrode heat generation is calculated using the equation presented in the theory:

$$Q_{el} = Q_{el,a} * (e^{Q_{el,b} * (1 - z/L)} - 1) \tag{5.21}$$

The coefficient $Q_{el,a}$ was calculated using furnace dimensions and furnace load to calculate the energy density and use it to calculate the coefficient $Q_{el,a}$ so that the energy added to the system equal the energy density at $z = 0$ m and decreases to zero at $z = 1$ m. We assumed that all the heat generated from the electrode is absorbed in the active area of the charge, an the calculations was done using following equation:

$$\text{Energy density} = \frac{\text{Furnace load}}{A_{ac,tot} * H_{hearth}} \tag{5.22}$$

where the energy density has units W/m^3, the furnace load is 30 000 W and H_{hearth} is 1 m. This leads to $Q_{el,a} = 28000$ W/m^3 The coefficient $Q_{el,b}$ was set to 5.

5.9 Thermal conductivity of solid and gas phase

In this section the calculations of thermal conductivity of the solid and gas phases will be explained. The aim is to generate temperature dependent equations for the thermal conductivity of each phase and hence each species. The approach to each species will we presented in addition to their sources. In the cases where an empirical equation is not presented in the theory, a temperature dependant function was made using the trend-line function in Excel, using existing data from various sources. These are shown in table 5.5.

The thermal conductivity of the gas phase, k_g (W/m·K), can be calculated by weighting the two existing gas species SiO and CO.

$$k_g = \sum_j x_{g,j} k_{g,j}, \quad j = SiO(g), CO(g) \tag{5.23}$$

Here $x_{g,j}$ is the mass fraction of species j in the gas phase. As thermal conductivity data for SiO gas is difficult to find, the thermal conductivity of the gas phase is calculated based on data for CO (g) taken from [37]. Excel was used to create a temperature dependant function from this data. As the $x_{SiO} + x_{CO} = 1$, k_g is expressed using data for CO gas. The final equation is presented in table 5.5. The thermal conductivity of the solid phase was weighted between all the solid and liquid species present as follows:

$$k_s = \sum_j x_{s,j} k_(s, j), \quad j = C(s), SiC(s), SiO_2(s), SiO_2(l), Si(l) \tag{5.24}$$

The mass fraction of species j in the solid phase is expressed as $x_{s,j}$. Kaisai et al. [39] obtained the following empirical equation for the determination of thermal conductivity k (W/m·K) as a function of temperature and porosity ε_C (-) of metallurgical coke using the laser flash method. This was done in the temperature range of 100 to 1400 °C. We will assume that the equation holds for our temperature domain. For these calculations the porosity of coke was set to 0.5.

$$k_{s,C} = (0.973 + 6.34 \cdot 10^{-3}(T_s - 273))(1 - \varepsilon_C^{\frac{2}{3}}) \tag{5.25}$$

Lezhenin and Gnesin [40] studied the thermal conductivity of silicon carbide, produced by reactive sintering, over the temperature range of 200 to 1650 °C. They expressed the temperature dependence of the thermal conductivity of silicon carbide in the form of an equation with units W/m°C, this was rewritten to equation (5.26).

$$k_{s,SiC} = 1.2 + 0,002(T_s - 273) + 0.62 \cdot 10^{-4} \cdot (T_s - 273)^2 \tag{5.26}$$

Thermal conductivity of liquid quartz is in this thesis set to be equal to solid quartz for simplicity. The conductivity was calculated based on thermal diffusivity measures from Ksiazek et al. [41], which measures thermal diffusivity of hydrothermal quartz and pegmatitic quartz. Both are used for metallurgical grade silicon production [41]. The measurements were done from room temperature to 1000 °C. These values were extrapolated to 1900 °C assuming that the thermal diffusivity of hydrothermal remained constant for the rest of the temperature interval, while the thermal diffusivity of pegmatitic quartz was extrapolated using a quartic polynomial.

An average was taken from the data at each temperature for the thermal diffusivities of the two quartz types. The data was then multiplied with the heat capacity of solid quartz and density taken from HSC [35]. The inbuilt excel trend-line function was then used to create a temperature dependant function presented in table 5.5.

Shanks et al. [42] measured thermal diffusivity of pure silicon from 300 to 1400 K and calculated the thermal conductivity of pure silicon in the same temperature range. The thermal conductivity of liquid silicon

is calculated based on the trend line of these data. The function is presented in table 5.5. The values used are presented in table 5.6.

Table 5.5: Thermal conductivity coefficients for CO gas, silicon and liquid and solid quartz in the function $k_i = c_{j1}T^4 + c_{j2}T^3 + c_{j3}T^2 + c_{j4}T + c_{j5}$ calculated using various sources.

species	$b_{j1}\ \frac{W}{mK^5}$	$c_{j1}\ \frac{J}{WK^4}$	$c_{j1}\ \frac{J}{WK^3}$	$c_{j1}\ \frac{W}{mK^2}$	$c_{j1}\ \frac{W}{mK}$
k_g	-1,00E-17	9,80E-11	-3,57E-08	9,13E-05	2,94E-04
$k_{s,Si}$	3,16E-09	-1,28E-05	0,0196	-13,676	4069.3
$k_{s,SiO_2(s,l)}$	8,79E-12	-4,26E-08	7,63E-05	-0,0594	19,4

Table 5.6: Thermal conductivity of silicon calculated by Shanks et al. [42]

Temperature K	**Thermal conductivity** W/cmK	**Temperature** K	**Thermal conductivity** (W/cmK)
300	1,422	900	0,337
400	0,974	1000	0,298
500	0,692	1100	0,29
600	0,577	1200	0,289
700	0,483	1300	0,288
800	0,4	1400	0,287

5.10 Specific heat of gas and solid phase

The specific heat of the solid phase, c (J/kgK) and the specific heat of the gas phase with constant pressure, c_p (J/kgK), for the solid and liquid phases are used in the energy balances for the gas and solid phase in order to calculate the phase temperatures. The heat capacity of each phase was calculated as a weighted mean of its components.

$$c_{p,g} = \sum_j x_j c_{p,j}, \quad j = SiO(g), CO(g) \tag{5.27}$$

$$c_s = \sum_j x_j c_{p,j}, \quad j = C(s), SiC(s), SiO_2(s), SiO_2(l), Si(l) \tag{5.28}$$

The heat capacity of each species was calculated using data from HSC [35] with temperature steps of 10 K between 298 K and 2773 K. The trend-line function was used to create a temperature dependant function for the capacity for each species as shown in equation 5.29. The coefficients used for each species are presented in table 5.7. For the solid and liquid species T_s was used to calculate the respective heat capacities, and T_g was used for the gas species.

$$c_{p,j}(T) = a_{j1}T^4 + a_{j2}T^3 + a_{j3}T^2 + a_{j4}T + a_{j5} \tag{5.29}$$

Table 5.7: Heat capacity coefficients for each species j, in the function $\Delta_f H_j(T) = b_{j1}T^4 + b_{j2}T^3 + b_{j3}T^2 + b_{j4}T + b_{j5}$. Data used to calculate the coefficient were taken from HSC [35]

species	$b_{j1}\ \frac{J}{K^4 mol}$	$b_{j1}\ \frac{J}{K^3 mol}$	$b_{j1}\ \frac{J}{K^2 mol}$	$b_{j1}\ \frac{J}{K mol}$	$b_{j1}\ \frac{J}{mol}$
$C(s)$	-1,948E-12	1,501E-08	-4,292E-05	0,05617	-4,595
$SiC(s)$	-2,858E-12	2,244E-08	-6,475E-05	0,08443	8,93
$SiO_2(s)$	-1,37E-11	8,93E-08	-2,03E-04	0,1957	2,567
$SiO_2(l)$	0	0	0	0	83,5
$Si(l)$	0	0	0	0	27,20
$CO(g)$	4,204E-13	-2,541E-09	3,557E-06	0,004230	27,325
$SiO(g)$	-5,632E-12	4,008E-08	-0,0001025	0,1161	7,694

Table 5.8: Molar mass of each species, values are taken from SI-chemical data [36]

species	$M_i \times 10^3\ \frac{kg}{mol}$
$C(s)$	12,01
$SiC(s)$	40,1
$SiO_2(s)$	60,09
$SiO_2(l)$	60,09
$Si(l)$	28,09
$CO(g)$	28,01
$SiO(g)$	44,09

5.11 Heat generation due to chemical reactions

In order to calculate the heat production due to chemical reaction, q''', calculation of reaction enthalpy of each reaction is needed:

$$q_s''' = \sum R_j \Delta H_{j,s}, \tag{5.30}$$

where i denotes the reaction and $\Delta H_{j,s}$ the reaction enthalpy contribution to the solid phase from reaction j. When a chemical reaction occurs, the standard enthalpy of reaction can be calculated by summing the standard enthalpy of formation of the reactants and subtracting the value of the sum of the standard enthalpy of formation of the products as shown below.

$$\Delta H_j = \sum \Delta_f H_{products} - \Delta_f H_{reactants} \tag{5.31}$$

Enthalpy of formation for each species i involved in the reaction can be calculated through:

$$\Delta_f H_i(T) = \Delta_f H_i^\circ + \int_{T_{ref}}^{T} C_{p,i} dT \tag{5.32}$$

where $\Delta_f H_i^\circ$ is the standard enthalpy of formation of species j at standard state pressure of 10^5 Pa and a temperature of 298 K. $c_{p,j}$ is the molar heat capacity at constant pressure for species i as a function of temperature. In this thesis however, the enthalpy of formation was calculated for each species using enthalpy data from HSC [35] with a temperature step of 10 K between 298 K and 2773 K to create a temperature dependant function using the trend-line function. Enthalpy data was used over heat capacity data because heat capacity data from HSC is prone to error as several of the species undergo several phase transformations in the chosen temperature interval 25-1850 °C. In HSC it is possible to choose between a Si or Si(l) "mode", which generates different results, as Si takes into account phase transformations. In this thesis, the Si(l) mode was used. The enthalpy functions are on the form of:

$$\Delta_f H_j(T) = b_{j1}T^4 + b_{j2}T^3 + b_{j3}T^2 + b_{j4}T + b_{j5} \tag{5.33}$$

The coefficients used to calculate the enthalpy of formation of each species is presented in table 5.9 and are chosen so that the units of $\Delta_f H_j$ are J/mol. A matrix has been made in order to present the data in a more organized matter: each species is given a number C (1), SIC (2) , SiO_2 (s) (3), SiO_2(l) (4), Si(5), CO (6) and SiO (7). So that a_{31} is the first coefficient in the formation enthalpy function for solid quartz.

Table 5.9: Enthalpy of formation coefficients for each species j, in the function $\Delta_f H_j(T) = b_{j1}T^4 + b_{j2}T^3 + b_{j3}T^2 + b_{j4}T + b_{j5}$. Data used to calculate the coefficient were taken from HSC [35].

species	$b_{j1}\ \frac{J}{K^4 mol}$	$b_{j1}\ \frac{J}{K^3 mol}$	$b_{j1}\ \frac{J}{K^2 mol}$	$b_{j1}\ \frac{J}{K mol}$	$b_{j1}\ \frac{J}{mol}$
$C(s)$	6,841E-10	-5,314E-06	1,607E2	2,528	-2224
$SiC(s)$	1,038E-09	-7,975E-06	0,02374	20,03317	-80032
$SiO_2(s)$	-3,707E-09	2,130E-05	-0,03319	86,726	-936747
$SiO_2(l)$	-3,707E-09	2,130E-05	-0,03319	86,726	-936747
$Si(l)$	0	0	0	27,2	40605
$CO(g)$	5,128E-11	-9,04797E-07	0,005021279	25,53211486	-118526,6155
$SiO(g)$	9,445E-10	-6,9374E-06	0,02078	30,274	-111621,0536

In this work the enthalpy contribution to each phase due to each reaction has been implemented differently to gain a stable framework and stable temperature profiles. For reaction R_1: SiO(g) + C(s) $\longrightarrow$ SiC (s) + CO(g), the enthalpy contribution to the gas and solid phase was chosen so that:

$$\Delta H_{1,g} = \Delta_f H_{CO}(T_s) - \Delta_f H_{SiO}(T_g) \tag{5.34}$$

and

$$\Delta H_{1,g} = \Delta_f H_{SiC}(T_s) - \Delta_f H_C(T_s) \tag{5.35}$$

Note that for gas products the solid temperature is used to calculate the formation enthalpy. For the remaining reactions the total reaction enthalpy was used together with coefficients to determine how much of the total reaction enthalpy should go to each phase. Note that R_2: SiO(g) $\longrightarrow$ Si(l) + SiO_2(l), and R_3 is the melting from solid quartz to liquid quartz, R_4: SiO(g) + SiC(s) $\longrightarrow$ Si(l) + CO(g) and R_5: SiO_2(l) + C(s) $\longrightarrow$ SiO (g) + CO(g). It has been assumed that the melting of quartz only affects the solid temperature.

$$q_g''' = -1\left(R_1 H_{1,g} + 0.1 R_2 \Delta H_2 + 0.3 R_4 * \Delta H_4 + 0.2 R_5 \Delta H_5 \right) \tag{5.36}$$

$$q_s''' = -1\left(R_1 H_{1,s} + 0.9 R_2 \Delta H_2 + R_3 * \Delta H_3 + 0.7 R_4 \Delta H_4 + 0.8 R_5 \Delta H_5 \right) \tag{5.37}$$

5.12 Reaction Rate Constants

In figure 7. in Sloman et.al [5] reaction rate and concentration of each species were plotted at different times. Reaction rates for time $t = 40$ mins, $t = 60$ mins and $t = 80$ mins was used to determine the reaction rate constants for the chosen reactions. Reaction rates for reaction 1, 2 and 4 the reaction rates vary in the interval 0-0.7 mol/m^{3}s. While for reaction 3 and 5 the reaction rates vary between 0-3.8 mol/m^{3}s. In order to determine the reaction rate constants it was chosen that the reaction rates for reactions 1, 2 and 4 are to be approximately 0.5 mol/m^{3}s while for reaction 3 and 5 to be 3.5 mol/m^{3}s at initial conditions. the reaction rate constant for reaction 5, k_5, was decreased to an order of 10^3 in accordance with Schei et al. [1]. The reaction rate will change as the concentrations of species, temperature and pressure changes. For k_2 is was chosen so that at a temperature interval between 25-1900 °C, the reaction rate R_2 was between 0 and 1. Concentration of SiO_2 (l) was set to 1E2 mol/m^3. Concentration of SiC(s) was set to 1E2 mol/m^3.

Table 5.10: Kinetic rates used in the model.

Kinetic rate constant	Value	Units
k_1	4.9E-10	sm/kg
k_2	9.8E-06	mol s /m^{2}kg
k_{-2}	0	m^3/mol s
k_3	5.9E-07	1/sK
k_4	1.1E-07	sm/kg
k_5	7.07E-08	m^3/mol s

Chapter 6

Verification Studies

As the proposed model of the low and high temperature zone of the silicon arc furnace is developed from first principles, a verification of its features is required, in order to determine whether or not the implementation of the model accurately represents the conceptual description of the problem. The main results in this thesis will be the temperature and concentration profiles from each case which are described by conservation of energy and species. In order to verify the model it is important to verify the source terms in the energy equation such as: h, q_s''' , q_g''' and Q_{el}, and the system when all the source terms are equal to zero. In addition, it is important to verify the mole balances and reaction heat generation. By validating these functions we obtain a better understanding of the functions and limitations of the models, which enables us to better understand the results obtained in case studies. This chapter will be divided into three sections: in section 6.1 species balance and reaction heat generation of each chemical reaction is investigated. In section 6.2 the temperature equation is investigated, as well as the effect of the heat exchange coefficient on the gas and solid temperature. The electrode heat generation is verified in section 6.3. Grid convergence was not evaluated in this work, and is an important further work for the developed framework.

The results presented in this chapter is based on an older version of the model, and do not correspond to the results in chapter 7. There have been several changes to the model, the main changes will now be mentioned. The molar velocities were set to constants, which stabilized the system drastically and allowed larger time steps to be used. The reaction rate constants were changed. The enthalpy used to be calculated based on heat capacity, now they are calculated based on enthalpy data from HSC [35], yielding a more correct calculation of reaction enthalpy (whether they are endothermic or exothermic) changes were also made in how the reaction heat was distributed between the gas and solid phase. During the simulations made in this section mole dependant variables were implemented as kmol in stead of in mol, this was changed. The constants used to calculate the electrode heat generation were fitted to the fictional furnace.

Even though the calculations done in this chapter are based on an older version of the model, the plots in this chapter illustrate the validation of the framework and potential information that may be obtained from the framework. Note that the low temperature zone in this chapter is 3 m - this was changed to 2.5 m for the final framework and the results presented in chapter 7.

6.1 Chemical reactions

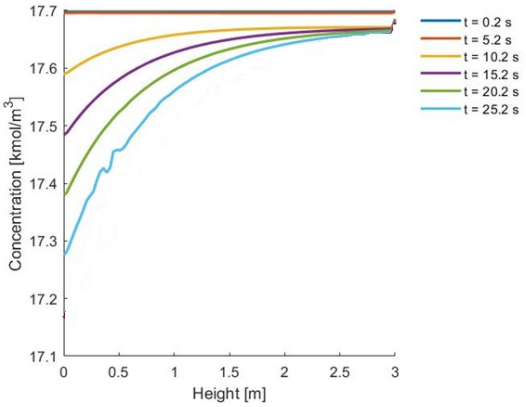
As species are consumed and created during the chemical reactions in this system, it is important to obtain mole balances for all the species. This is done by only allowing a single reaction to run at a time by setting all but one of the reaction coefficients to 0 and plotting the concentration profiles of the involved species in the reaction to see if the mole balance is conserved during simulations. The following section will therefore investigate if the mole balance is conserved in each reaction. Each reaction rate is defined such as in the theory: it is the reaction coefficients k_i that will be set to either 0 or 1E-7 during this case study. The chemical reactions are either endothermic or exothermic and it is important to verify the source terms q_s''' and q_g''' as they represent the heat production due to the chemical reactions. Due to time limitations the verification will be limited to checking whether the reactions are endothermic or exothermic by plotting the temperature profile of the solid and has phase when allowing only one and one reaction to run, some times the source terms q_s''' and q_g''' are used.

6.1.1 Reaction 1: Species balance

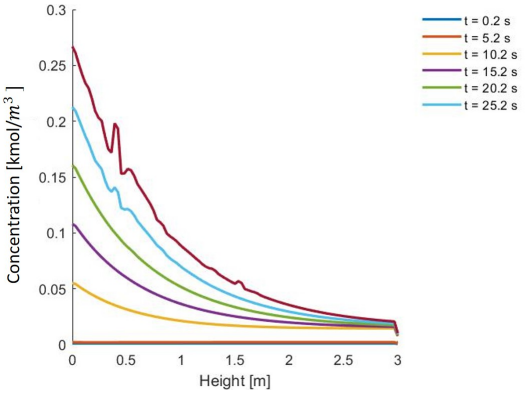
Species balance of reaction 1 was tested by setting k_2 to k_5 equal to zero and $k_1 =$ 1E-7 ms/kg. Running the simulation and then plotting the concentration profiles of the involved species in reaction R_1 at different times. Reaction R_1 is as presented in the theory:

$$\mathrm{SiO\,(g)} + 2\,\mathrm{C\,(s)} \xrightarrow{\mathrm{R_1}} \mathrm{SiC\,(s)} + \mathrm{CO(g)}, \tag{6.1}$$

It is therefore expected that the production of SiC is half that of the consumption of carbon. We expect to observe an increase in CO(g) concentration and a decrease in SiO(g) concentration due to this reaction. It is also expected that the partial pressure of CO(g), P_{CO}, increases as carbon-monoxide is produced, while it is expected that the partial pressure of SiO(g) will decline as it is consumed. This should also correlate well with the reaction rate. Therefore the concentration profile of C_C and C_{SiC} was plotted and compared. The partial pressures P_{SiO}, P_{CO} and the reaction rate R_1 are also presented.

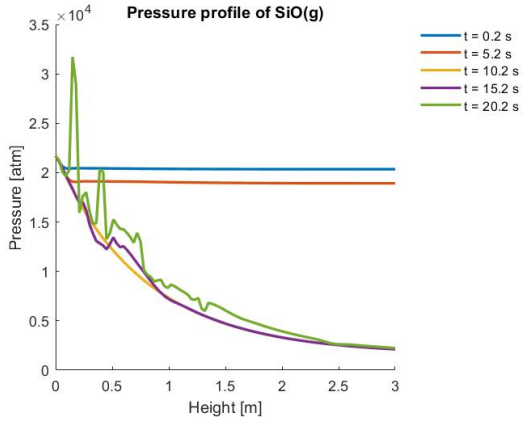


(a) Concentration of carbon.

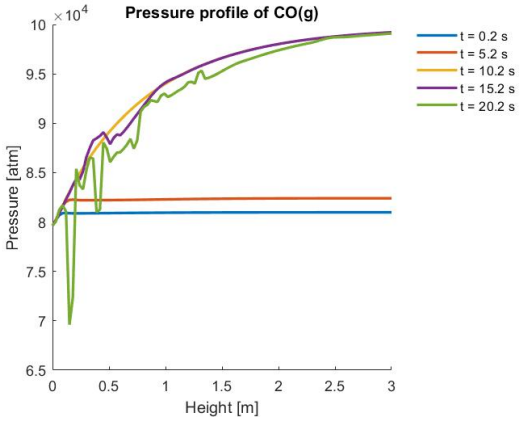


(b) Concentration of SiC(s)

Figure 6.1: Species balance for reaction R_1



(a) Pressure profile of SiO(g)



(b) Pressure profile of CO(g)

Figure 6.2: Gas partial pressures for reaction R_1

As seen in figure 6.1a and 6.1b C(s) is consumed. Around twice as many moles of carbon is consumed as SiC is produced. It is therefore possible to observe that the species balance of C(s) and SiC(s) is conserved. In figure 6.2a the partial pressure of SiO(g) decreases while the partial pressure of CO(g) increases in figure 6.2b. This matches our expectancy as the SiO(g) is consumed in the reaction and CO(g) is produced. After a certain amount of time C_{CO} and C_{SiO} are very unstable, due to numerical oscillations. This was fixed in the present work.

6.1.2 Reaction 1: Enthalpy balance

Reaction R_1 is exothermic. In order to validate this the temperature profile, in addition to calculated ΔH_{rx1} is plotted in the following figures. If the reaction is exothermic, we expect to see an temperature increase in the solid and gas phase temperature as the reactions runs. The solid temperatures were initialized at $T = 25$ °C, while the gas phase had a gradient between 1800 and 25 °C.

$$\mathrm{SiO\,(g)} + 2\,\mathrm{C\,(s)} \xrightarrow{\mathrm{R_1}} \mathrm{SiC\,(s)} + \mathrm{CO(g)}, \quad \Delta\mathrm{H} = -77.592\,\mathrm{kJ/mol} \tag{6.2}$$

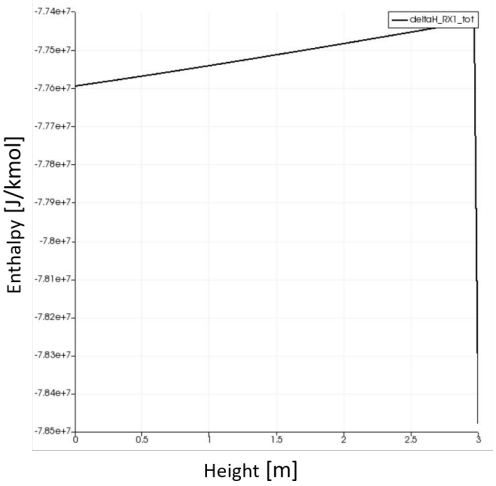
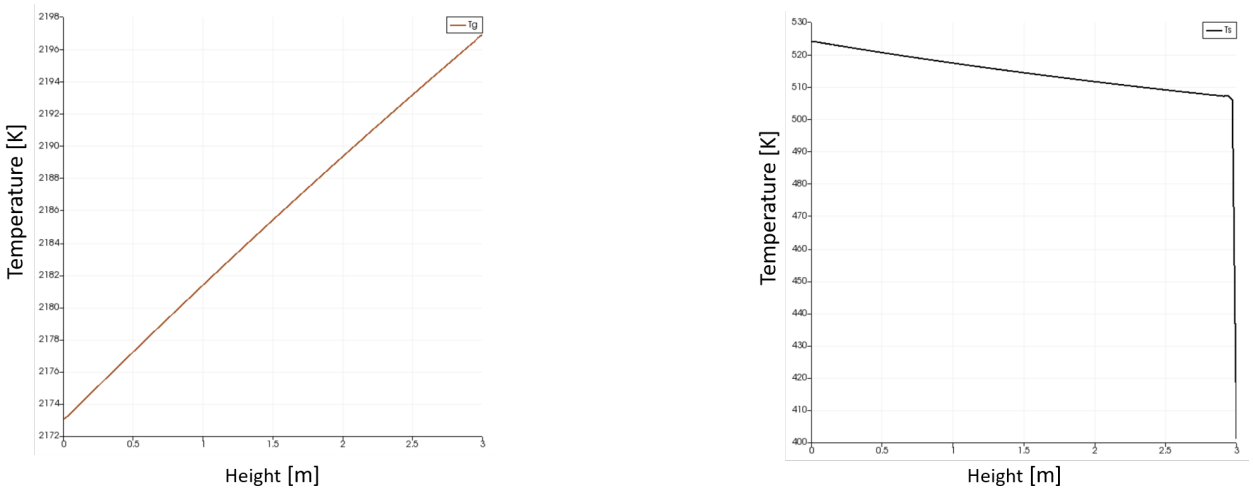


Figure 6.3: Total reaction enthalpy for reaction 1 in the entire domain of the LT model at $t = 6$ s

As seen from figure 6.3, the reaction is exothermic. In figures 6.4a and 6.4b it can be observed that the temperature of the solid and gas phase increase according to the exothermic reaction.



(a) Temperature profile of gas phase

(b) Temperature profile of solid phase

Figure 6.4: Temperature profiles due to reaction 1

6.1.3 Reaction 2: Species balance

The mole balance of reaction R_2 was investigated by setting all reaction rate constants to 0 except k_2. The case was run and the concentration profiles of the involved species in reaction R_2 were plotted. Reaction 2 is as presented in the theory:

$$2\,\mathrm{SiO\,(g)} \xrightarrow{\mathrm{R_2}} \mathrm{Si\,(l)} + \mathrm{SiO_2(l)}, \tag{6.3}$$

As the condensation reaction of SiO(g) runs it expected that the partial pressure decreases, as the concentration of Si(l) and SiO_2(l) increases. It is also expected that the increase of concentration of the liquid species are equal, as they are one to one in the reaction. This should also correlate with the calculated reaction rates at those time steps. Therefore the concentration profile of C_{Si} and $C_{SiO_2(l)}$ were plotted, in addition to pressure profile P_{SiO} in addition to the reaction rate R_2. They were all plotted at different times to show the dynamic changes. In figure 6.5 We see that the reaction rate of R_2 becomes zero after around 0.7 s. At t=0.1 s the reaction rate is 2.25E-3 kmol/s. And the concentration profile of Si(l) and SiO_2 are both 2.25E-4 at the same position at the same time. In which makes sense as the concentration only 0.1 s has passed an the reaction rate is describes as per second. The reaction rate becomes zero as the equilibrium partial pressure for SiO gas is highly temperature dependent and the reactions stop when the gas temperature becomes higher than about $T_g = 1850$ °C.

Concentration profile of Si(l) and SiO_2(l) are shown in figure 6.6a and 6.6b respectively. The concentration profiles of these two species match as expected. They however stabilize around $t = 0.2$ s and seemingly stop running as $R_2 = 0$ at $t = 0.7$ s in figure 6.6a. In figure 6.7a the pressure profile of SiO(g) was plotted and is decreasing. However the pressure profile is moving towards the right even though the profile is constant, this is expected due to the rapid movement of the gas phase and that the reaction rate R_2 becomes 0 after 0.2 s.

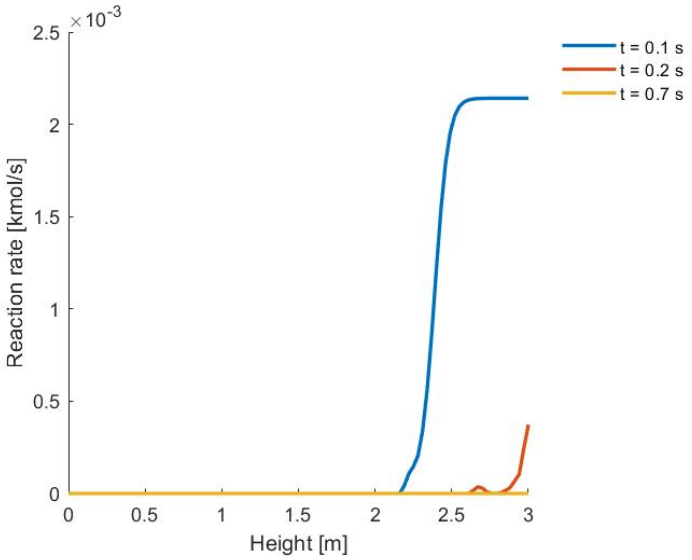
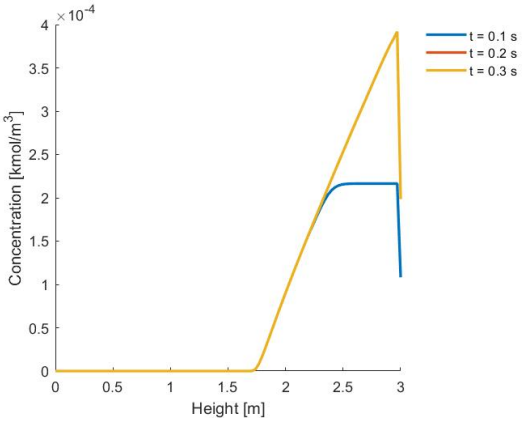
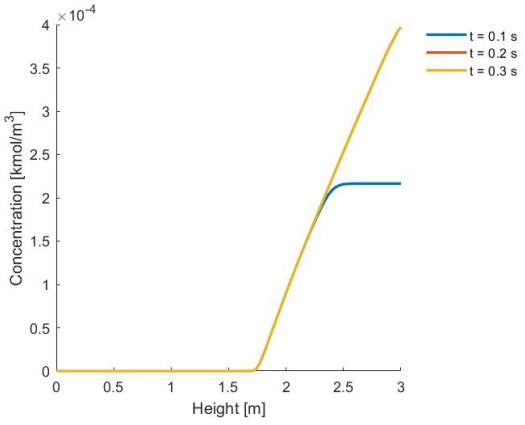


Figure 6.5: Reaction rate R_2

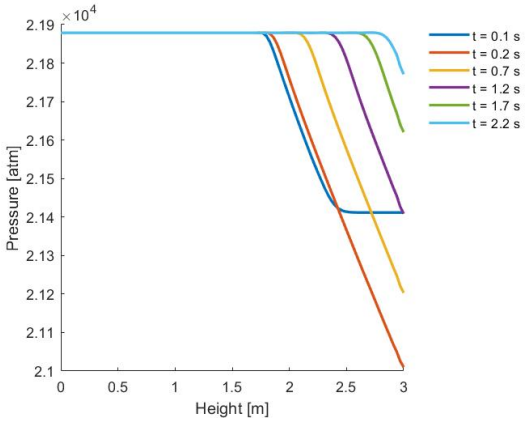


(a) Concentration profile of Si(l)

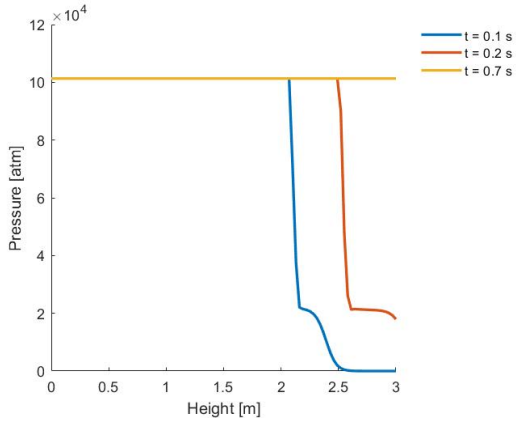


(b) Concentration profile of SiO_2(l)

Figure 6.6: These figures show the concentration generated from only allowing reaction R_2 to occur.



(a) Pressure profile of SiO(g)



(b) Pressure profile of $P_{SiO,equilibrium}$

Figure 6.7: Partial pressure of SiO(g) and equilibrium pressure $P_{SiO,equilibrium}$ for reaction R_2.

6.1.4 Reaction 2: Enthalpy balance

Reaction R_2 is a condensation reaction of $SiO(g)$ and runs as follows discussed in the theory:

$$2\,\mathrm{SiO\,(g)} \underset{\mathrm{R_{-2}}}{\overset{\mathrm{R_2}}{\rightleftharpoons}} \mathrm{SiO_2(l)} + \mathrm{Si(l)}, \quad \Delta\mathrm{H}{=}{-}306\,\mathrm{kJ/mol} \tag{6.4}$$

As this is an condensation reaction which is exothermic it is expected to see an temperature increase in the system. However the calculated reaction enthalpy is positive, according to our model, which is wrong. This was due to an error in the equations, which was changed in the present framework. Changes in the temperature in the gas were small due to the reaction as shown in figure 6.9b, while the temperature in the solid phase was unaffected as shown in figure 6.10b. The simulation was initialized with a temperature gradient from $T_g = 1800$ °C to $T_g = 25$ °C, but the boundary inlet was set to $T_{g,inlet} = 1900$ °C.
$\dot{q}_s'''$ is negative when the reaction runs, which makes sense at the reaction enthalpy calculated is endothermic, but $\dot{q}_g'''$ is positive. which means that the temperature in the gas phase increases. This was changed in the current work.

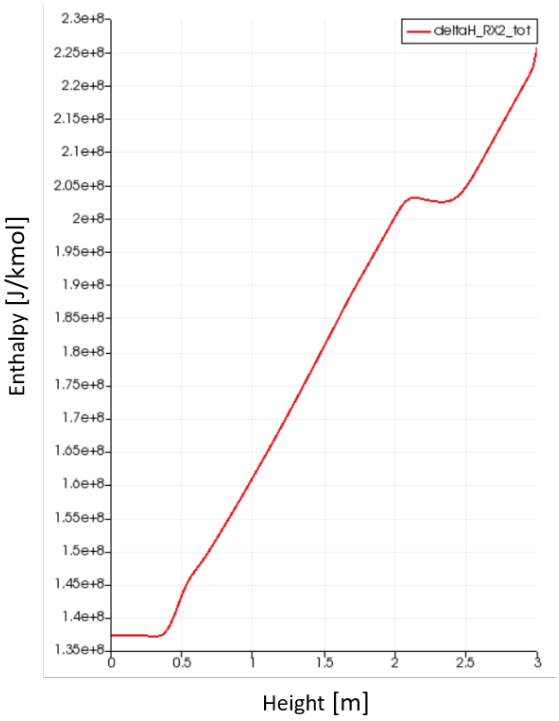
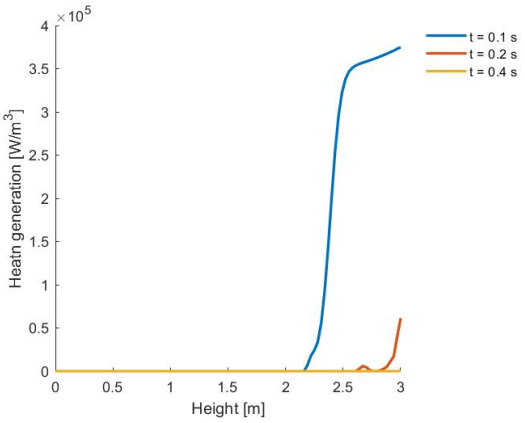
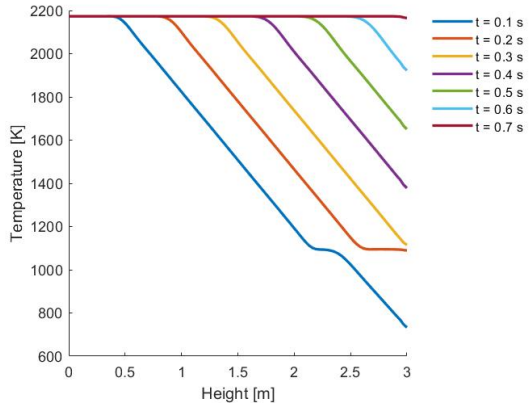


Figure 6.8: Reaction enthalpy for reaction R_2.

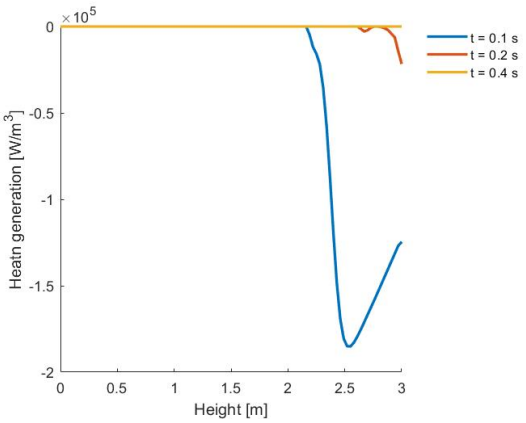


(a) Heat generation in the gas phase

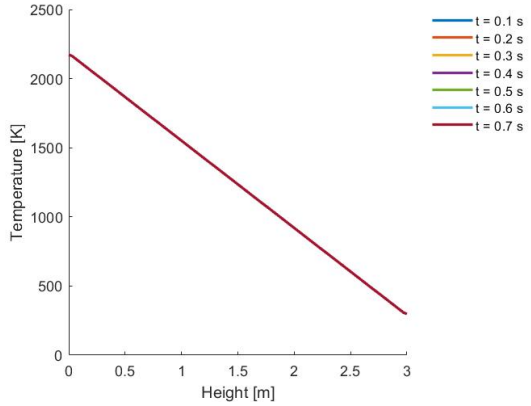


(b) Change in temperature profile of gas phase due to reaction heat

Figure 6.9: The effect of reaction R_2 on the temperature profile heat generation in the gas phase.



(a) Heat generation in solid phase



(b) Temperature profile of solid phase

Figure 6.10: The effect of reaction R_2 on the temperature profile of the solid phase, and heat generation

6.1.5 Reaction 3: Species balance

The solid and gas phase were initialized with an temperature gradient from $T = 1800$ to 25 °C.

$$\mathrm{SiO_2(s)} \xrightarrow{\mathrm{R_3}} \mathrm{SiO_2(l)}. \tag{6.5}$$

As seen in figures 6.12a and 6.12b the mole balance is conserved. However only at elevated temperatures, which makes sense as the melting temperature of quartz is $T_m = 1723$ °C (1996 K). But the reaction rate R_2 is quite small, 4E-5, when the reaction coefficient is set to 1E-7.

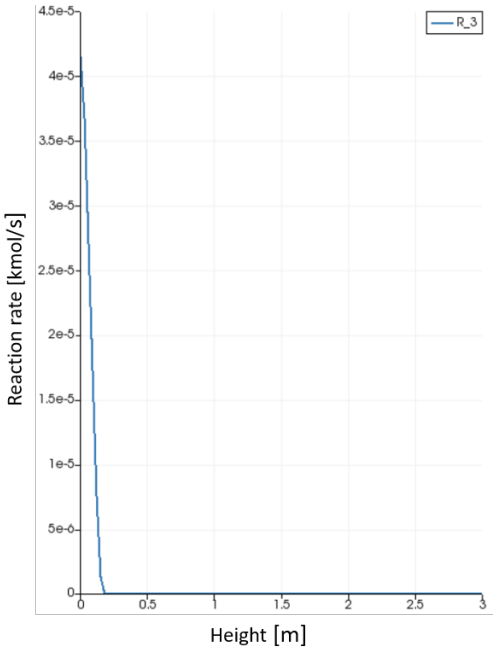
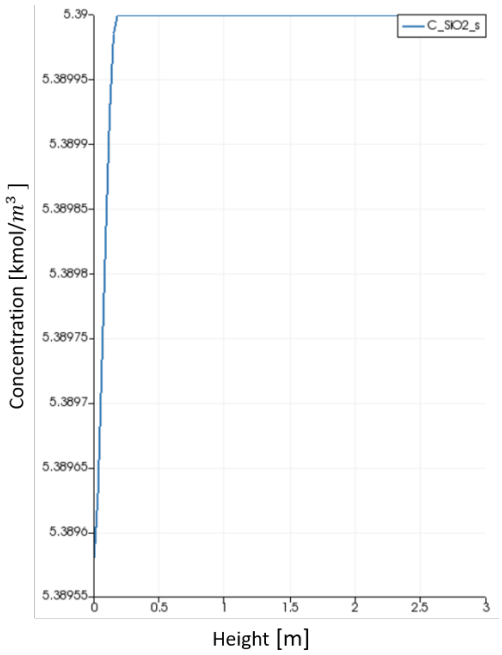
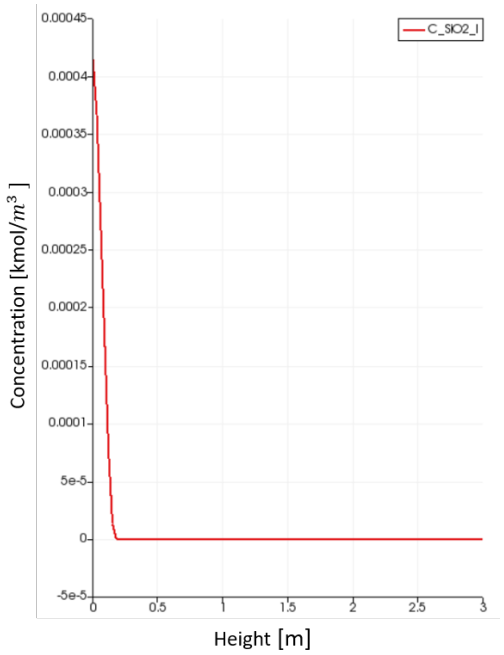


Figure 6.11: Reaction rate R_3.



(a) Concentration of SiO_2(s)



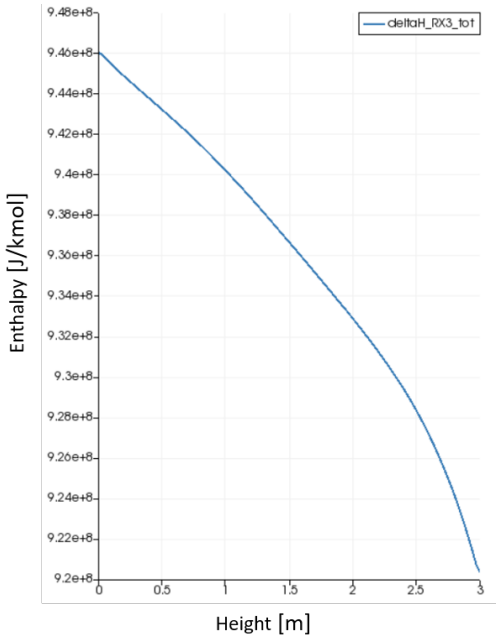
(b) Concentration of SiO_2(l)

Figure 6.12: The effect of reaction R_3 on concentration profiles of involved species after 10 s

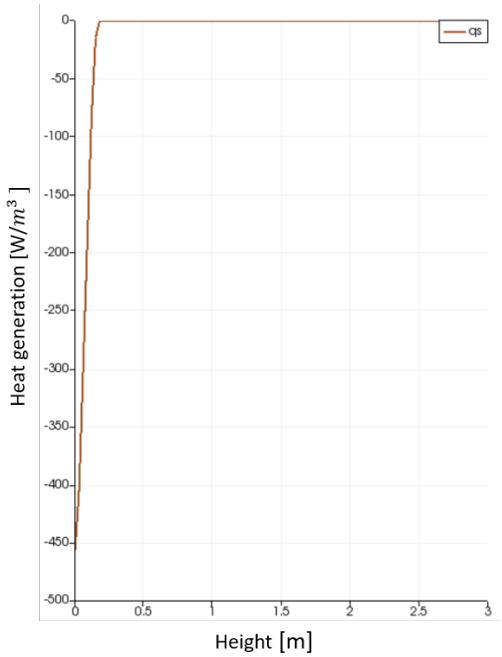
6.1.6 Reaction 3: Enthalpy balance

The calculated total reaction enthalpy for reaction R_3 is endothermic as presented in figure 6.13a. This also causes the solid temperature to decrease where the reaction takes place. Hence q_s''' is negative as presented in figure 6.13b. The temperature profiles were not affected by the reactions, mainly because the reaction rate was so small, so they were not presented here.

$$\mathrm{SiO_2(s)} \xrightarrow{\mathrm{R_3}} \mathrm{SiO_2(l)}. \tag{6.6}$$



(a) Total reaction enthalpy for reaction R_3 in the entire domain of the LT model



(b) Heat generation in solid phase

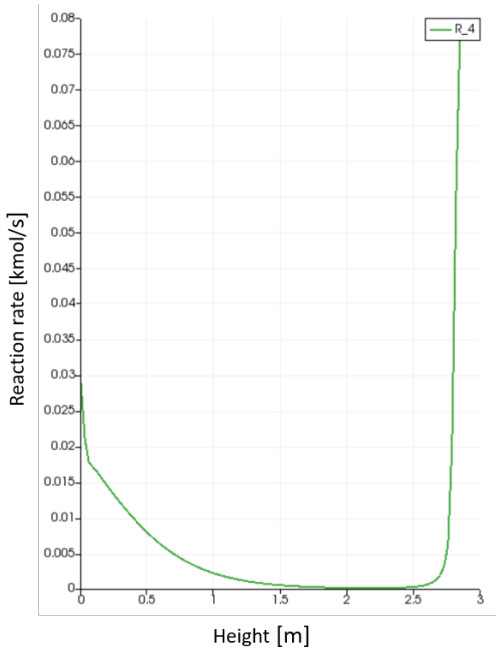
Figure 6.13: Results from reaction R_3

6.1.7 Reaction 4: Species balance

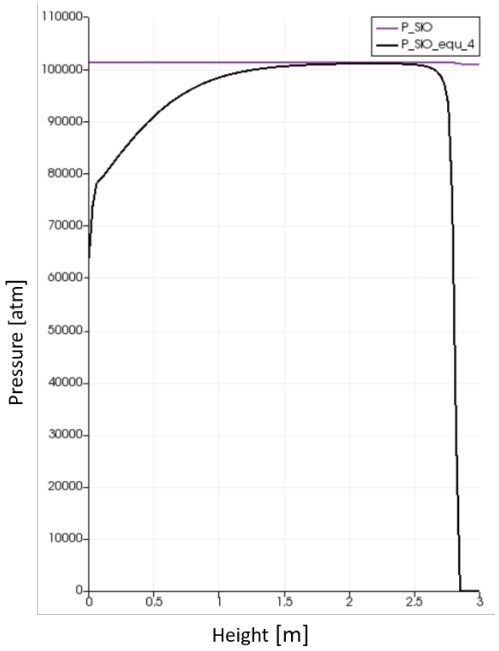
Species balance of reaction R_4 was tested by setting $k_1 - k_3$ and k_5 equal to zero. For this case we have to initialize the system with concentration of SiC(s) as 7.7 kmol/m^3 and the inlet partial mole fraction of SiO gas to be high, almost equal to 1, as this reaction requires high partial pressure of SiO gas in order to occur. All these results were plotted at $t = 0.003$ s. This is because the framework stopped running due to instabilities in the temperature profile.

$$\mathrm{SiO\,(g) + SiC\,(s) \xrightarrow{R_4} 2\,Si\,(l) + CO(g),} \tag{6.7}$$

From the figures below we can observe species balance for the gas and the solid species.

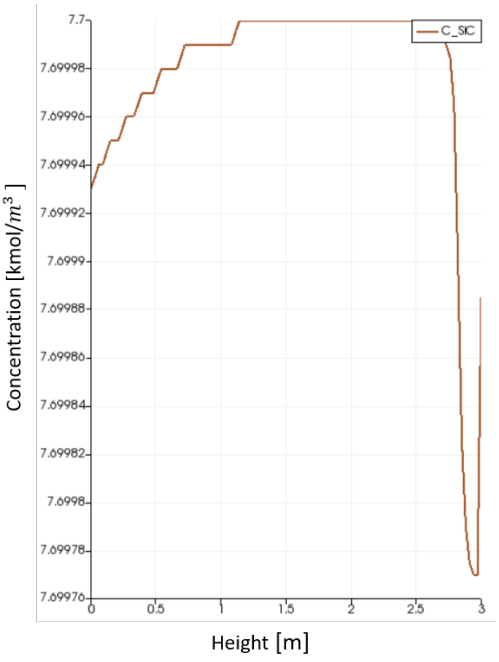


(a) Reaction rate for reaction 4

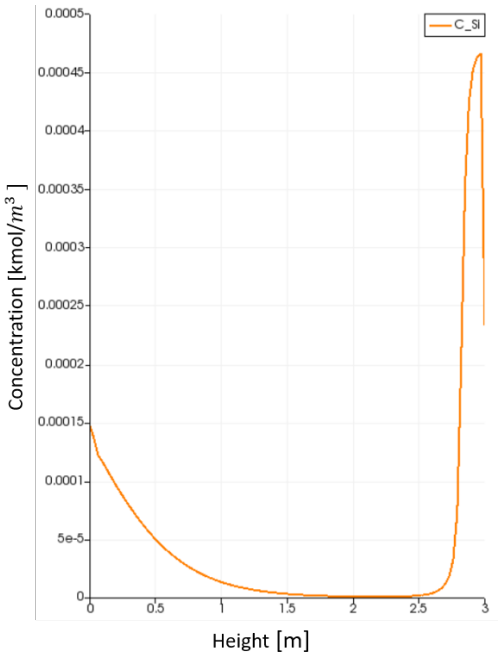


(b) Pressure profile for SiO gas and equilibrium SiO

Figure 6.14: link between reaction rate and equilibrium pressure for SiO g

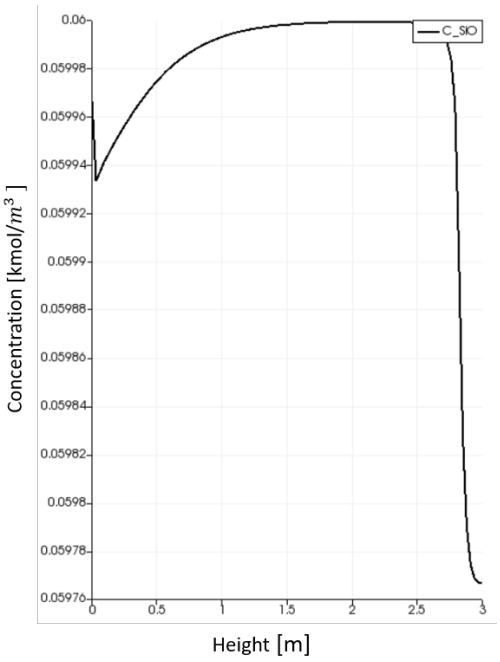


(a) Concentration of SiC solid

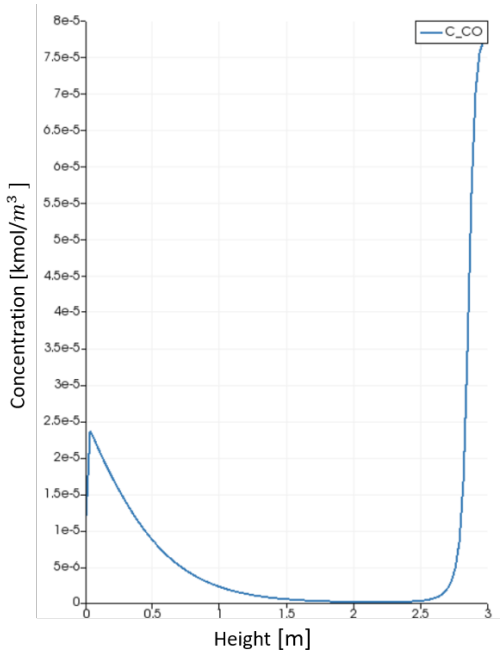


(b) Concentration of liquid Si

Figure 6.15: Concentration profiles of reaction R_4 at time 0.003 s



(a) Concenctration of SiO gas



(b) Concentration of CO gas

Figure 6.16: Concentration profiles of reaction R_4 at $t = 0.003$ s

6.1.8 Reaction 4: Enthalpy balance

As presented in the theory, reaction R_4 is endothermic. The total calculated reaction rate (shown in figure 6.17) was also endothermic in the system domain. q_g''' and q_s''' graphs are of the same over the entire domain, and are both negative. T_g drops visibly where the reaction rate is high. However, the case simulations stopped after $t = 0.003$ s. This is probably due to the temperature drop of the gas profile at $z = 3$ m. This is most probably due to the high reaction rate R_4 see figure 6.14a due to the high difference between the partial pressure of SiO(g) and equilibrium pressure of SiO gas presented in figure 6.14b. This might be because the equilibrium pressure for SiO(g), drops to zero when the temperature of the gas is to low, below $T_g = 600$ °C. All the plots are made at $t = 0.003$ s.

$$\mathrm{SiO\,(g)} + \mathrm{SiC\,(s)} \longrightarrow 2\,\mathrm{Si\,(l)} + \mathrm{CO(g)}, \quad \Delta\mathrm{H} = 166.71\,\mathrm{kJ/mol} \tag{6.8}$$

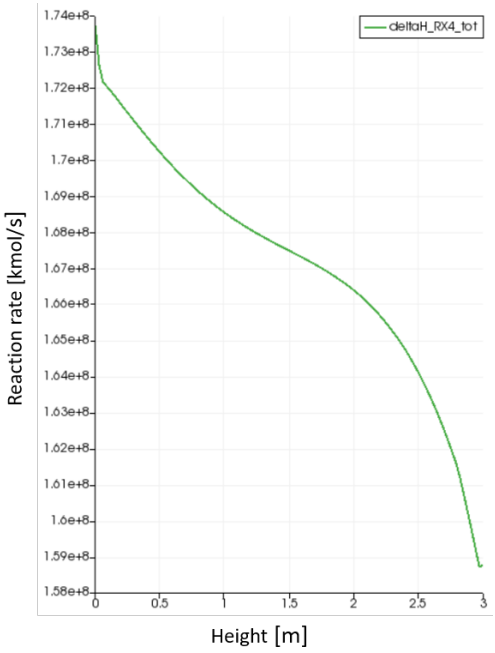
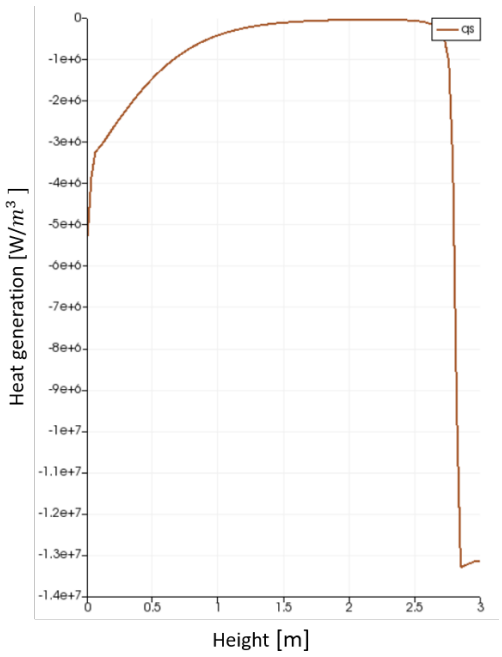
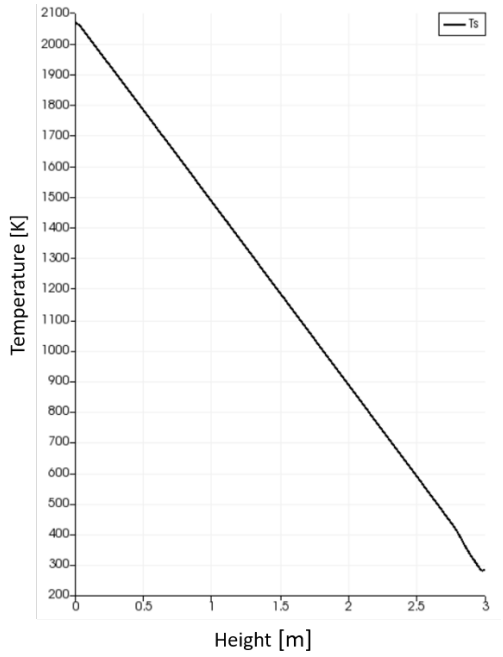


Figure 6.17: Total reaction enthalpy for reaction R_4 in the entire domain of the LT model

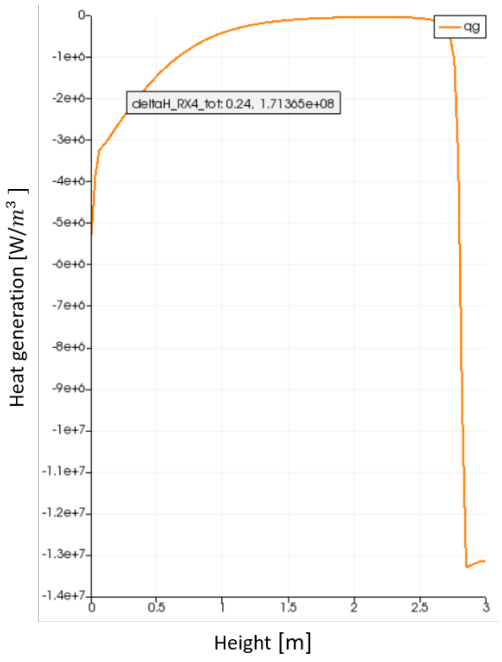


(a) Heat generation in solid phase

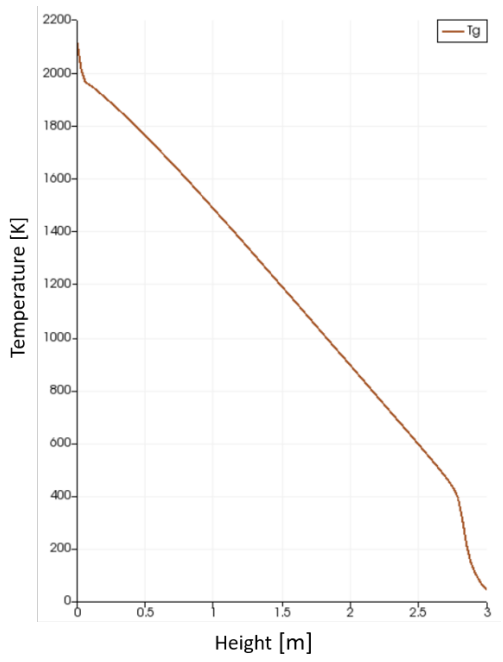


(b) Temperature profile of solid phase

Figure 6.18: The effect of reaction R_4 on temperature profile of solid phase and heat generation



(a) Heat generation in gas phase



(b) Temperature profile of gas phase

Figure 6.19: The effect of reaction R_4 on temperature profile of solid phase and heat generation

6.1.9 Reaction 5: Species balance

The reaction rate $R_5 = 3.54$E-5 kmol/m^{3}s, so the effect on the species concentration was not able to be observed as shown in figure 6.22a and 6.22b probably due to the low reaction rate. We could however observe production of CO(g) and SiO(g). The gas concentrations seem unstable in addition to having low values and only minor effects on partial pressure of CO and SiO gas. This case was run with high partial pressure of CO(g), $C_{CO} = 0.0581$ mol/m^{3}and $C_{SiO} = 0.016$ mol/m^3. All plotted graphs are plotted at $t = 3$ s.

$$\mathrm{SiO_2(l) + C\,(s) \xrightarrow{R_5} SiO\,(g) + CO\,(g)} \tag{6.9}$$

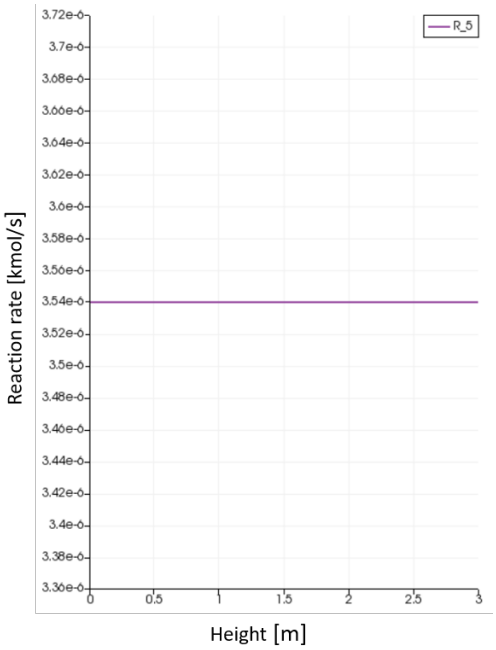
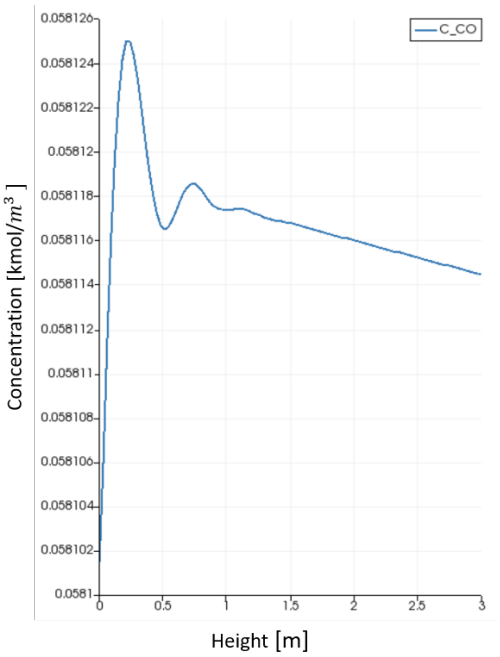
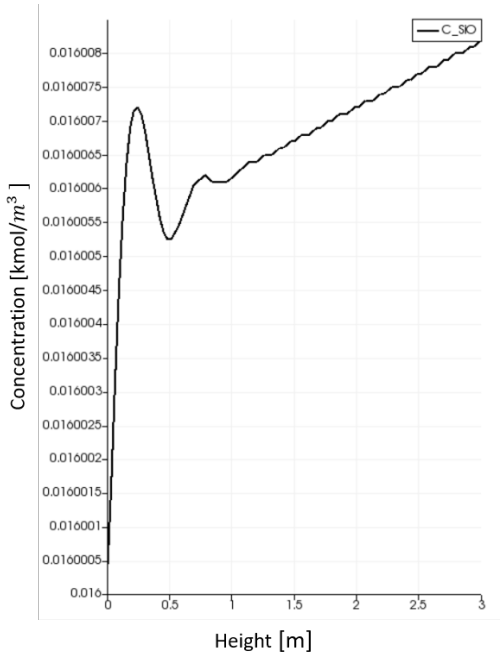


Figure 6.20: Reaction rate R_5

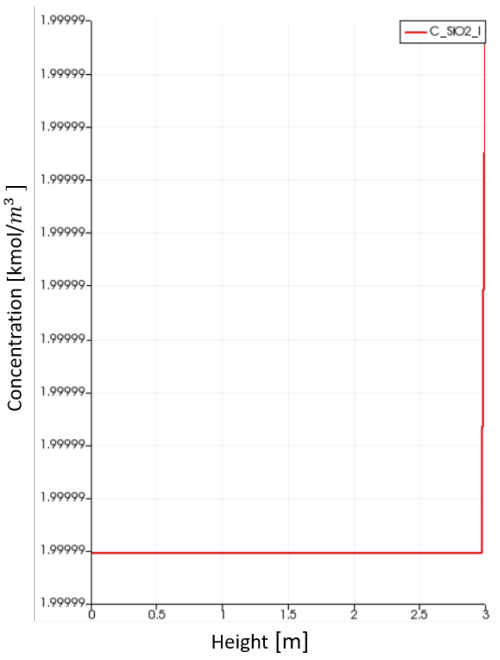


(a) Concentrations of CO gas

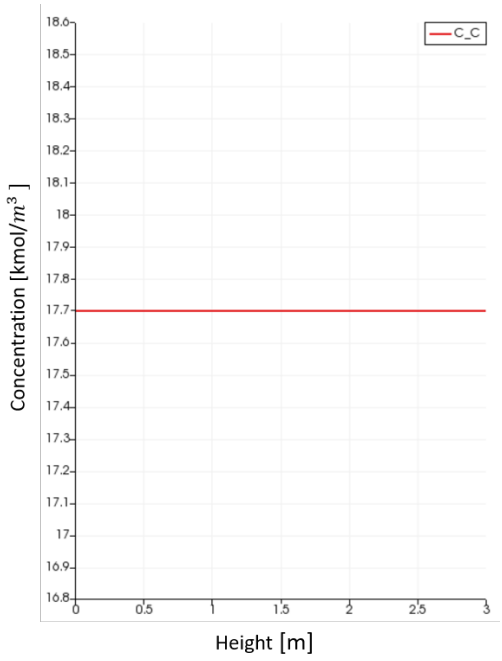


(b) concentrations of SiO gas

Figure 6.21: The affect of reaction R_5 on concentrations profiles for CO and SiO gas



(a) Concentration of SiO_2(l)



(b) concentrations of C(s) gas

Figure 6.22: change in concentration of SiO2(l) and C(s) due to reaction R_5

6.1.10 Reaction 5: Enthalpy balance

As presented in the theory, this reaction is expected to be endothermic, however the calculated total reaction enthalpy is exothermic ranging from -5E8 to -2E8 J/kmol. In addition it does not seem to affect the temperature of the solid phase, but a small increase of the temperature in the gas phase. The low temperature change due to reaction R_5 is probably due to low reaction rates and thus low production due to the chemical reaction. The enthalpy calculations for reaction R_5 were changed in present work see section 5.11.

$$\mathrm{SiO_2(l) + C(s) \longrightarrow SiO\,(g) + CO(g), \Delta H = 792.510\,kJ/mol} \tag{6.10}$$

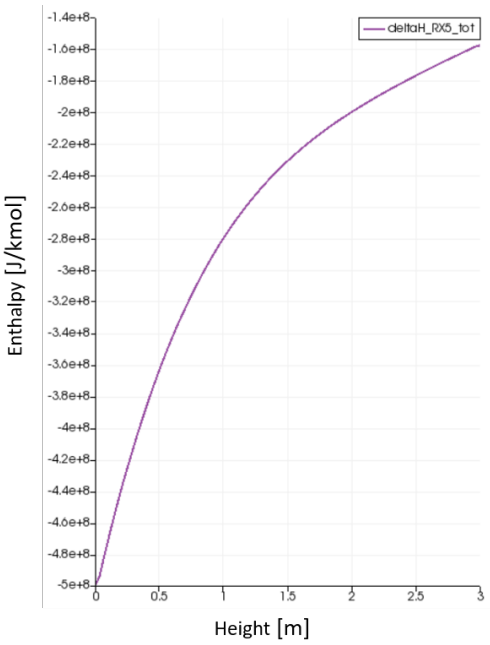
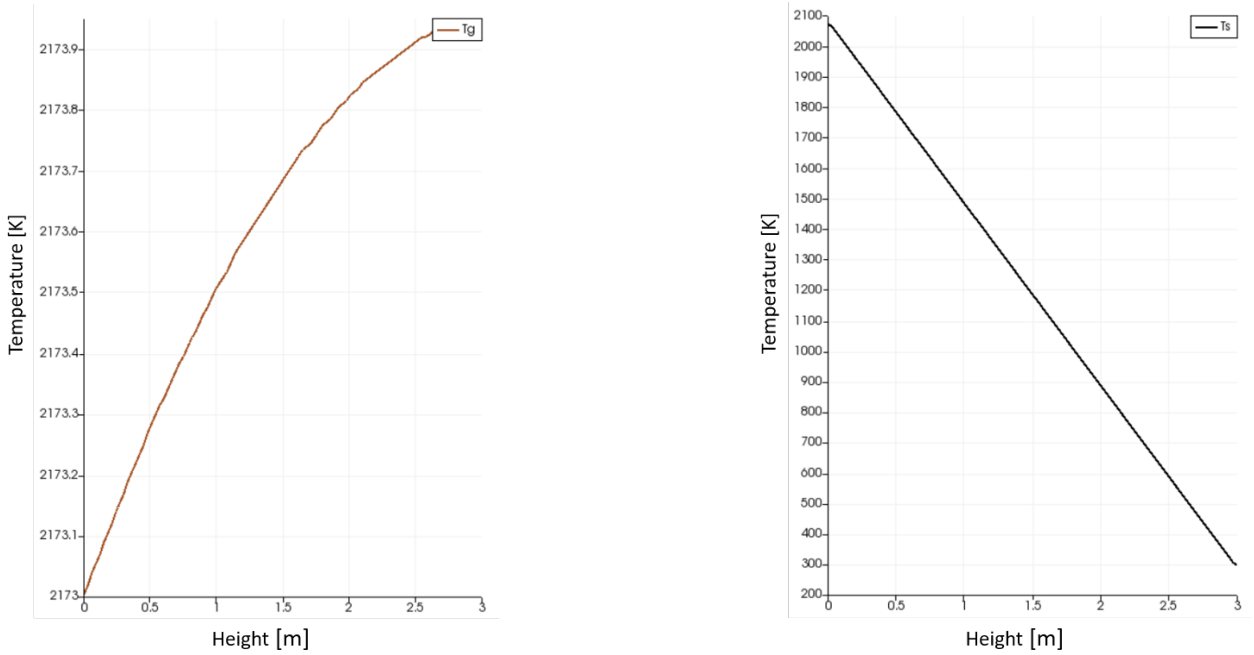


Figure 6.23: reaction enthalpy for the entire reaction



(a) Temperature of gas phase

(b) Temperature of solid phase

Figure 6.24: Temperature profile change due to reaction R_5.

6.2 Verification of the temperature equation

In order to verify the functionalities of both the energy conservation equation and its calculations of the temperature profile, in addition to the heat exchange coefficient and the source term from the electrode heat generation the chemical reactions are deactivated in order to see their affects individually. This section will therefore not focus on reaction heat generation, as q_s''' and q_g''' are subsequently zero. The temperature equation is as stated in the theory:

$$(1-\varphi)(\rho c)_s \frac{\partial T_s}{\partial t} + v_{z,s}(\rho c)_s \frac{\partial T_s}{\partial z} = (1-\varphi)q_s''' + h(T_g - T_s) + (1-\varphi)Q_{el} \tag{6.11}$$

$$\varphi(\rho c_p)_g \frac{\partial T_g}{\partial t} + v_{z,g}(\rho c_p)_g \frac{\partial T_g}{\partial z} = -h(T_g - T_s) + \varphi Q_{el} + \varphi q_g''' \tag{6.12}$$

The boundary conditions for the temperature in the gas and solid phase is presented in the table 6.1 below, while the gas and solid velocity are presented in table 6.2. The gas and solid velocity are calculated based on the base case. All the cases in this subsection will use these conditions unless anything else is stated. The initial temperature distribution inside the model domain is set as a linear temperature gradient from $T = 2173$ K at the inlet to $T = 298$ K at the outlet of the model for both the gas and solid temperature.

Table 6.1: The boundary conditions for the temperature in the solid and gas phase

T	BC at inlet [K]	BC at outlet[K]
T_g	2173	298
T_s	298	2173

Table 6.2: The gas and solid velocity used during verification of the temperature equation

Velocity	Value [m/s]
U_g	1.29
U_s	5.25E-5

6.2.1 Temperature equation: No heat exchange

In order to verify the model it is important to investigate if the model can calculate the temperature distributions when initial species are present with no chemical reaction, deactivation of all source terms and no heat exchange between the phases. The energy balance equation was re-implemented as:

$$(1-\varphi)(\rho c)_s \frac{\partial T_s}{\partial t} + v_{z,s}(\rho c)_s \frac{\partial T_s}{\partial z} = h(T_g - T_s) \tag{6.13}$$

$$\varphi(\rho c_p)_g \frac{\partial T_g}{\partial t} + v_{z,g}(\rho c_p)_g \frac{\partial T_g}{\partial z} = -h(T_g - T_s) \tag{6.14}$$

Where $h = 0$. Since there are no source terms and no heat exchange between the phases, it is expected that the temperature profiles reaches to their boundary conditions after a certain amount of time. It is expected that the temperature profile of the gas reaches $T_{g,boundary} = 1900$ °C as it stabilizes. Since the $U_g = 1.29$ m/s this should take around 2.3 seconds. The solid gas temperature profile is however expected to reach $T_{s,min} = 298$ K as the temperature of the solid at the inlet is 298 K and the solid has the opposite direction of the gas. As the velocity of the solid is $U_s = 5.25$E-5 m/s it is expected to take around 57000 s. The temperature profile of the solid and gas phase was therefore plotted in figure 6.25 and figure 6.26 below in order to investigate this.

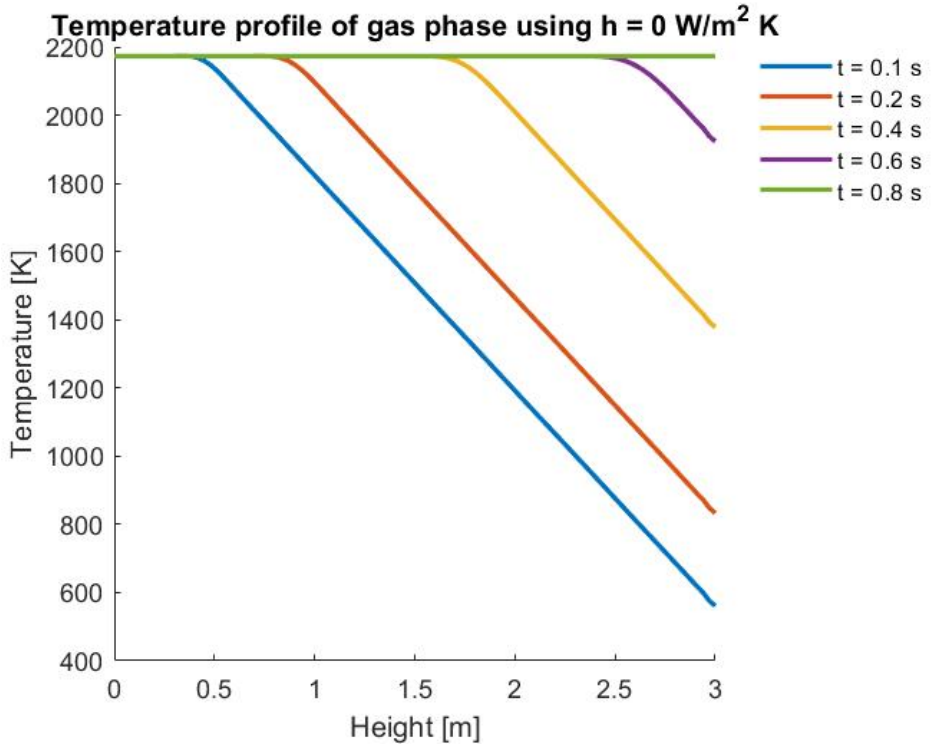


Figure 6.25: Temperature profile for the gas phase at different times with no heat exchange between the phases or energy sources.

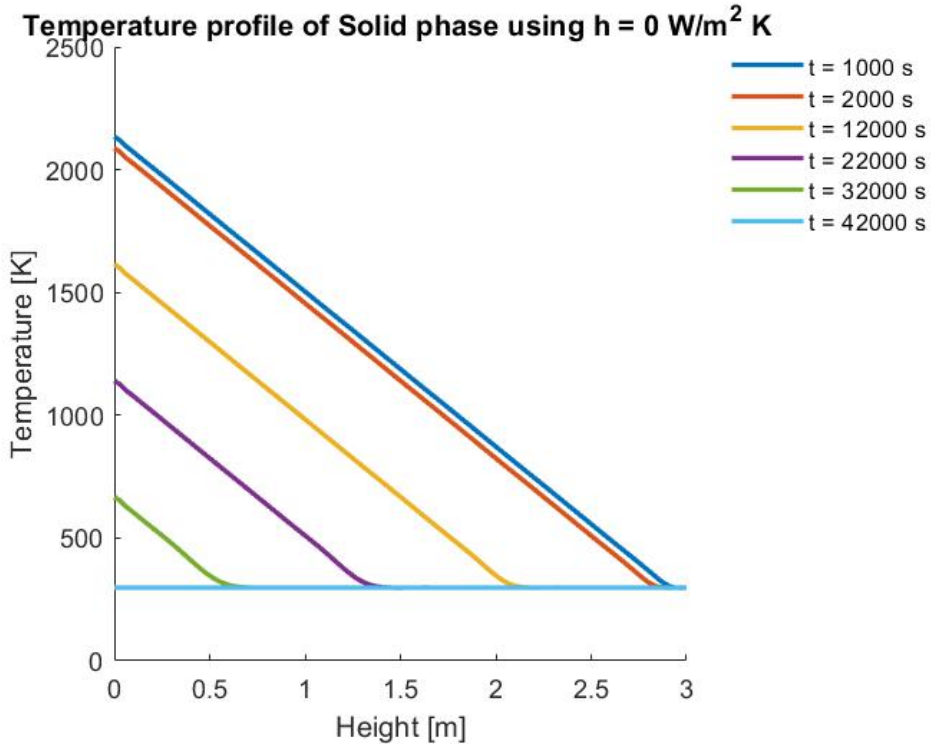


Figure 6.26: Temperature profile for the solid phase at different times with no heat exchange between the phases or energy sources.

When the temperature profile of the gas phase is plotted it only takes around 0.8 seconds for the temperature inside the entire domain to reach the maximum temperature. This deviates from the expected time estimate. However, as discussed in the theory, the temperature profile moves with the average velocity in stead of the superficial velocity. The average velocity is $V_g = \frac{U_g}{\varphi} = 4.34$ m/s. The time it should take for the entire temperature profile to reach the maximum temperature is then 0.69 s. When plotting only that time step, it can be observed that the bend is there. So it might be that the velocity is correct, but due to numerical error, the entire temperature profile is not entirely 2173 K. However, the temperature of the gas does not exceed the maximum temperature of the gas, which makes sense, as all the source terms have been deactivated.

In the solid phase the average solid velocity is $V_s = \frac{U_s}{1-\varphi} = 7.5$E-5 m/s. The solid temperature profile should then reach T_{min} after 40000 s. The temperature profile of the solid phase however reaches $T_{s,min}$ in the entire domain at around 42000 s, which deviates from the expected time. The temperature profile of the solid stays within the expected boundaries and stabilizes at the inlet temperature.

6.2.2 Temperature equation: Heat exchange

As the heat exchange coefficient will be used throughout the case studies presented in the following chapter, its functionality is essential in order to obtain sensible results. One simple case was computed in order to validate the source term. As the heat exchange coefficient is set to be dependant on $k_s(T_s)$ and $k_g(T_g)$, it is expected to also be temperature dependant. In order to investigate this the heat exchange coefficient will be plotted with temperature. All the other source terms have been deactivated and the reaction rate constants have all been set to 0 so there are no heat production due to reactions.

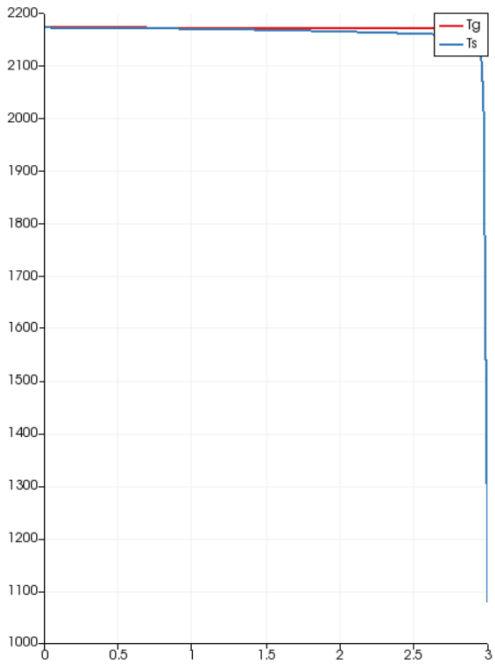


Figure 6.27: Temperature profile for the gas and solid phase at $t = 304$ s

In figure 6.27 The temperature of the solid and gas phase are presented. It shown that the gas and solid temperature seem to overlap in the domain and the gas temperature at $z = 3$ m is 2200 K.
In figure 6.28 we can observe that the calculated h has a size between 55 and 3 W/m^{3}K. In figure 6.29 h^* ranges from 1.1 to 0.06 W/m^{2}K. From calculations in MATLAB using $T = 2173$ K the result h is of order 10^5, this was corrected in present work, see section 5.7.

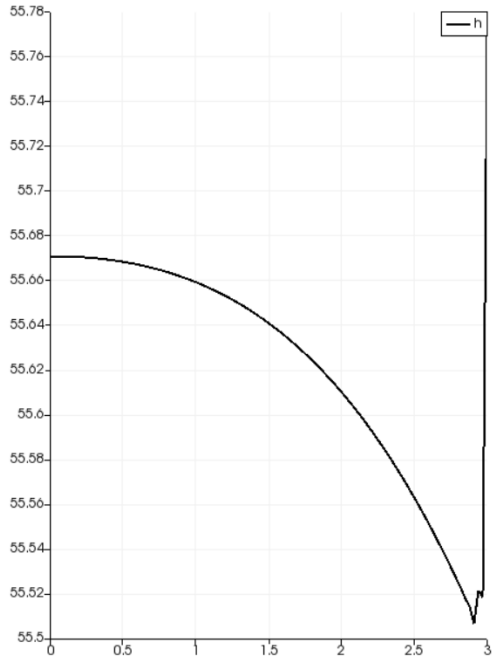


Figure 6.28: Heat exchange coefficient (W/m^{3}K) profile for the gas and solid phase at $t = 304$ s

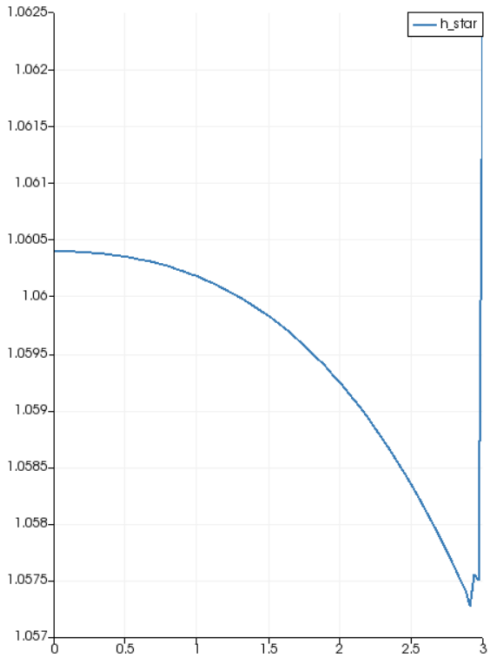


Figure 6.29: Actual Heat exchange coefficient (W/m^{2}K) profile for the gas and solid phase at $t = 304$ s

6.3 Electrode heat generation

As the submerged electrodes stands for about 45% of the energy consumption in order to generate high temperature inside the furnace to produce molten silicon the electrode heat generation is an important parameter as a heat source and will be used in the case studies presented in the following section. The electrode heat source term will be validated through comparing to its analytic solution.

The velocity of the solid phase was set to zero. It only contributes as a heat source for the solid temperature. Or else the gas temperature will increase drastically. This was changed in present work. The electrode heat generation is set to be expressed by:

$$Q_{el} = A(-1 + \exp(B(1 - \frac{z}{L}))) \tag{6.15}$$

Where $A = 200$ kg/ms^3 and $B = 5$ are parameters set in by the user and z denotes the z-position. In figure 6.30 is can be observed that the calculated electrode heat generation from OpenFOAM corresponds to the analytical solution.

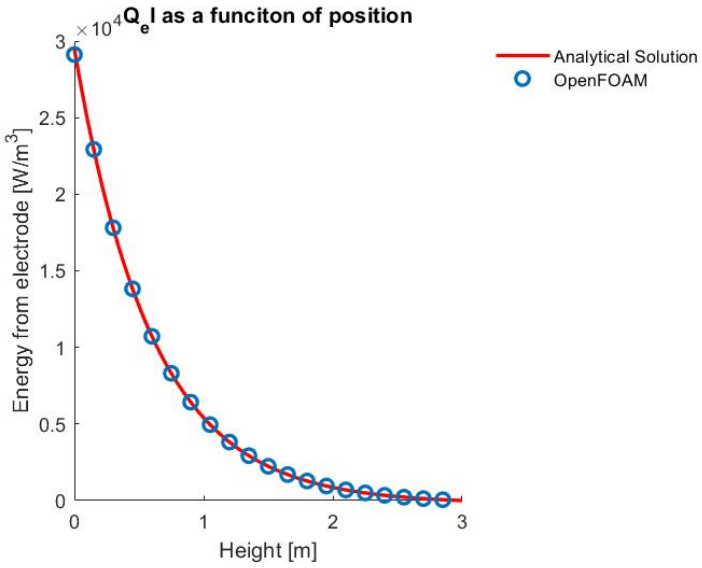


Figure 6.30: Generated energy from the electrode as a function of temperature. Comparing analytical solution from MATLAB and results from OpenFOAM

Chapter 7

Results

In order to investigate how different parameters affect the dynamics of the modelled zones in the furnace, three cases will be evaluated and presented in this chapter. The three cases are 1) Change in carbon reactivity 2) Concentration of carbon entering the low temperature zone and 3) Change in SiC concentration entering the high temperature zone. As the model still is too simple to capture the actual physics of the environment inside the furnace, such as conduction of heat in each phase, heat transport due to radiation or diffusion to walls or neighbouring charge, the aim is not to compare the model to industrial cases as this would require extensive work on the models, the aim is to focus on dynamic changes in the temperature and concentration profiles to observe if these models can capture such behaviour and if this is motivation enough to continue with development of the model in order to get accurate temperature profiles from an industrial furnace in order to utilize it to maximize the profit of the waste heat from the furnace off-gas.

7.1 Base case LT-and HT-model

The base case for the low temperature zone used the calculated values from chapter 5. The reactivity of carbon, r_C, is set to 0.1 in the base case for the low temperature zone. According to theory, the gas temperature at the outlet from the low temperature model should be around 1400 ° C [1]. It is also expected that the carbon has mainly reacted to SiC(s), and that sufficient amounts of quartz has melted to liquid quartz. The results of the base case for the low temperature model has been presented in section 7.2.

In the base case for the high temperature model the height has been adjusted to 1 m, and assumed that 70% of the carbon has reacted with SiO(g) to SiC(s) according to reaction R_2. So that the system is initialized with 30% C(s) and 70% SiC(s) (the inlet composition has been set to the same). Results from the base case from the high temperature model is presented in section 7.4. In the high temperature model it is expected to have high concentration of liquid silicon, and to see a significant rise in reaction rate R_4 and R_5 compared to the low temperature model as there is heat generation from the electrode in this zone raising the temperature and liquid quartz and SiC(s) is present in the system.

7.2 Effect of change of reactivity of carbon

As mentioned in the introduction, the carbon reactivity, r_C, is one of the most important parameters for furnace operation. The aim of this case is therefore to investigate the affect the carbon reactivity has on the gas temperature profile inside the furnace. Changing the reactivity of carbon can be thought of as changing the carbon material. This would affect the reaction rate, R_1 as presented in table 3.1 for reaction $\mathrm{SiO(g)} + 2\,\mathrm{C(s)} \longrightarrow \mathrm{SiC(s)} + \mathrm{CO(g)}$. The reaction reactivity was implemented as a constant in the framework, and is set to 0.1, 0,2, 0,3 and 0,5 in order to investigate the effect on the gas temperature profile in the LT-model. This means that 4 cases were initialized, and run in 22 h (80400 s) with time steps of 1 s. The four cases took around 318 s to complete the calculations. As the solid velocity is 6.25E-5 m/s it should take around 40000 s for the charge to go through the entire domain. The cases are run for 80400 s so that the charge runs through the system twice, in order to assure stability in the system.

Reaction R_1 is exothermic, and it is therefore expected that the temperature of the gas should increase for higher carbon reactivity. With a higher carbon reactivity it is also expected to observe higher SiC concentrations in the furnace. We want a systematic way of measuring how furnace behaviour changes with varying inputs of carbon reactivity and define a SiC yield for this case as following:

$$\text{SiC Yield (thousandth)} = \frac{\text{Total SiC content at each position}}{\text{Total input carbon}} \times 1000 \tag{7.1}$$

which is measured in terms of concentrations. This calculates the amount of SiC that created for reaction between solid carbon and SiO gas. The higher conversion of C(s) to SiC(s) the better. Considering the uncertainties of kinetic parameters and the simplifications done in this framework these parameters are not indented to portrait a high level of accuracy, but are useful to compare for different values of carbon reactivity. The reaction rate constants have been observed to be a lot lower than in industrial furnaces, so the results cannot be directly transferred to operational setting. With concentrations of SiC in the system it is expected to observe higher reactivity for R_4, SiO gas reacting with SiC(s), which is an strongly endothermic reaction.

In figure 7.1 the temperature profiles of the gas phase using different values for the carbon reactivity has been presented when the system has stabilized. The model indicates that a lower carbon reactivity gives a lower temperature in the upper part of the low temperature zone. When the system has stabilized at $t = 1340$ min, the gas temperatures at the outlet vary as shown in table 7.1 from 1687-1739 K (1414-1466 °C). The

gas temperature profile and gas temperature at the outlet increase as expected. It seems that the reactivity of carbon has the highest effect on the gas temperature profile at $z > 0.5$ m from figure 7.1.

Table 7.1: Temperature of the gas at the outlet ($z = 2.5$ m) when system has stabilized with different reactivities of carbon

Carbon reactivity, r_C	**$T_{g,out}$[K]**
0.1	1687
0.2	1701
0.3	1715
0.5	1739

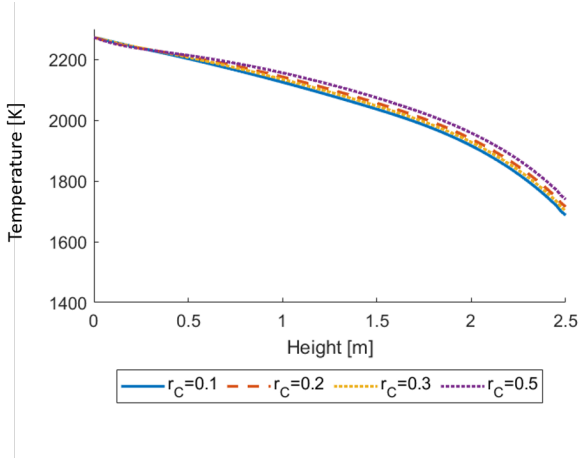
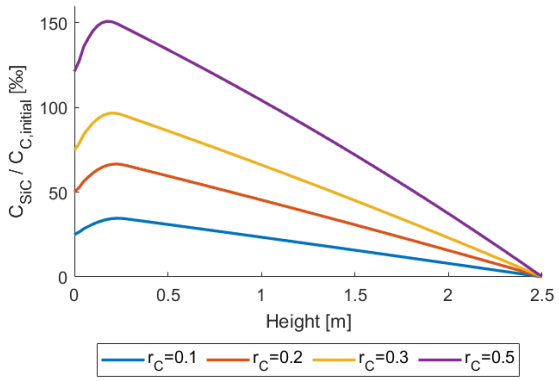
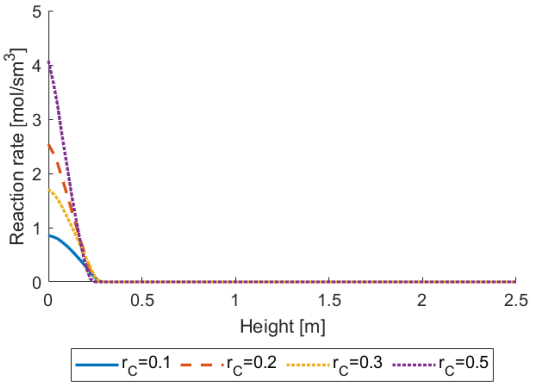


Figure 7.1: Temperature profile of the gas phase using different carbon reactivities.

In figure 7.2a the SiC yield was plotted as a function of height (z) with different carbon reactivities when the system has stabilized. The higher reactivity number indicates a higher SiC yield. The SiC yield varies from 25 to 80‰. There is a drop of SiC yield in the lower parts of the model ($z < 0.5$) in all the subfigures. This coincides with increasing reaction rate for reaction R_4 (SiO + SiC $\longrightarrow$ 2 Si + CO) as shown in figure 7.2b. The graph indicates the huge effect of the carbon reactivity number which enhances the Si-production.



(a) The figure shows the SiC yield with different reactivities of carbon.



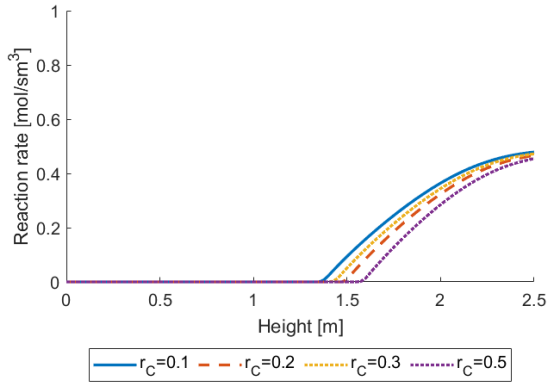
(b) Reaction rate R_4 profile after $t = 100$ min.

Figure 7.2: Reaction rate, R_4, and SiC yield for four different values of carbon reactivity after $t = 1340$ min

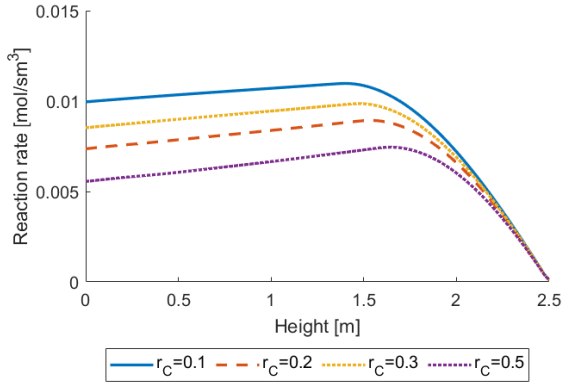
From figure 7.3a the SiO condensation reaction R_2 ($2\,SiO(g) \longrightarrow Si(l) + SiO_2(l)$) take place at the upper 1.5 m and decreases with higher carbon reactivity. From figure 7.3b the direct reaction $SiO_2\,(l) + 2\,C\,(s) \longrightarrow SiO(g) + CO(g)$ is very low varying between 0.005 and 0.01 mol/m^{3}s. The reaction is controlled by a low reaction rate in accordance with Schei et al. [1]

The graph in figure 7.4b shows how the concentration of the SiO gas in the low temperature zone is affected by changing reactivity. The concentration of SiO(g) decreases with increasing carbon reactivity, however the concentration seem to vary only a little with the height. In figure 7.4a the graph shows the concentration of the liquid silicon with different carbon reactivity. The flat curve from 0.2 to 1.7 m is due the condensation reaction as seen from reaction rate R_2 in figure 7.3a. The high concentration from 0 to 0.2 m is due to the Si production as seen from reaction rate R_4 in figure 7.2b. A higher carbon reactivity gives a higher concentration of Si(l) below 0.2 m, however, a higher reactivity gives a lower concentration of Si(l) from 0.2 to 1.7 m.

In figure 7.5 the melting of quartz ($SiO_2(s) \longrightarrow SiO_2(l)$) is indicated to take place between 0.2 m to 0.5 m of the low temperature zone. The findings of Nordnes [43] are shown in figure 7.5b for comparison. Note that in figure 7.5 the bottom of the furnace is at $z = 0$ m, while in figure 7.5 the bottom is at -2.5 m.

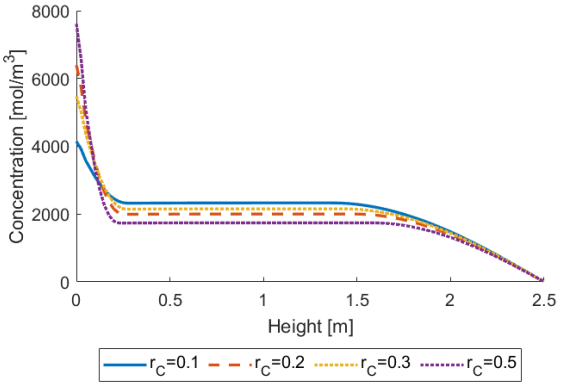


(a) Reaction rate R_2 for four different values of carbon reactivity.

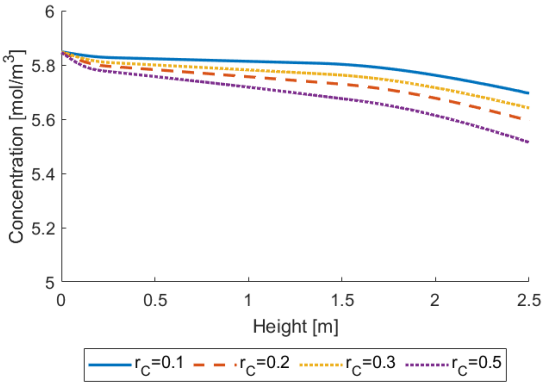


(b) Reaction rate R_5 for four different values of carbon reactivity.

Figure 7.3: Reaction rate R_2 and R_5 for four different values of carbon reactivity

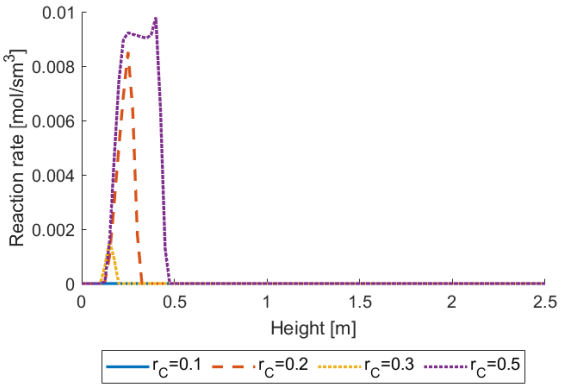


(a) Concentration of Si(l) with different carbon reactivities

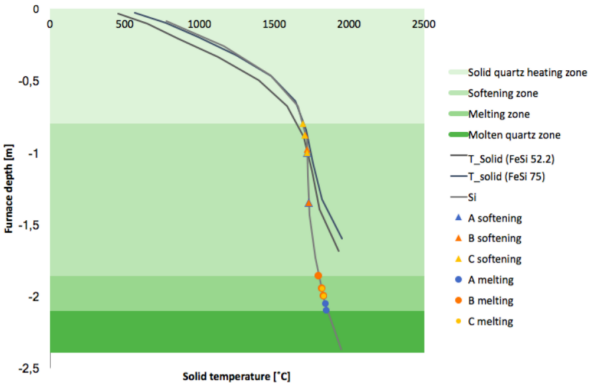


(b) Concentration of SiO(g) with different carbon reactivities.

Figure 7.4: Concentration of SiO2(l) and Si(l) with different carbon reactivities.



(a) Reaction rate R_3 for four different values of carbon reactivity.



(b) This figure is taken from Nordnes [43] and illustrated the temperature profile of solids in production furnace and apparent softening and melting temperature from sessile drop experiments are marked on the graph for silicon.

Figure 7.5: Reaction rate R_3 for melting of quartz calculated in this model compared to results from Nordnes [43]

7.3 Effects of change in charge composition

As mentioned in the introduction, according to Schei et al. [1], the charge composition is one of the parameters a metallurgist can change. The framework created allows to change the charge composition while the furnace is running. The aim in this case is to illustrate how this affects the the concentration and temperature profiles inside the low temperature area. The inlet temperatures will be kept constant in addition to gas composition. The solid phase temperature will be initialized as a temperature gradient from 2000 °C at $z = 0$ m to 25 °C at $z = 2.5$ m. The temperature of the gas at the inlet is set to $T_{g,t,inlet} = 2000$ °C. This case will be run by firstly running the LT- model using the base case with carbon reactivity $r_C = 0.2$ from $t = 0$ until $t = t_1$, where the system has stabilized. We let the the case run until the charge has changed two times, so $t_1 = 80400$ s. Then at $t = t_1$ the charge composition is changes to either under or over coked until the system is filled with the new charge or stabilizes. As the solid velocity is 6.25E-5 m/s it should take 40000 s for the new charge to go though the entire domain, but in order to run the case until is has stabilized the cases are run for 60000 s, total time is then 140400 s. The calculations done when changing the charge took around 256 s in execution time for each case.

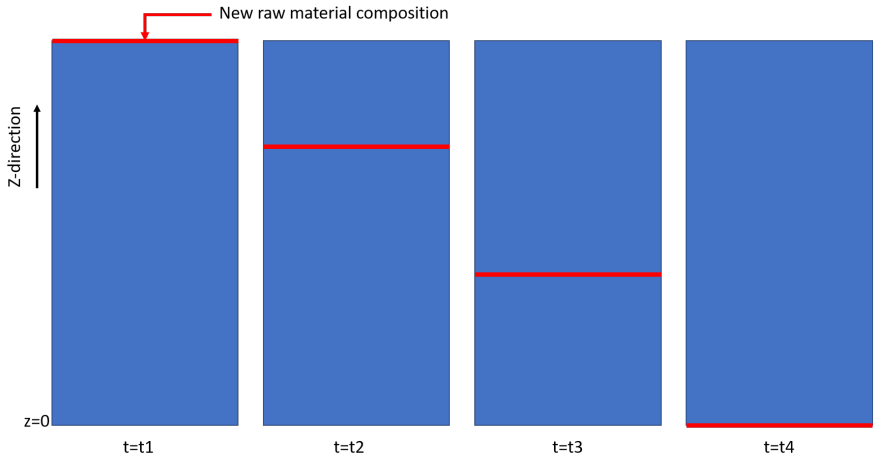


Figure 7.6: The figure explains how the new charge composition goes through the model at different times.

Table 7.2: Input values for the input charge material composition in mole fraction

Case	$C_{C(s)}$
+50%	106125
+10%	77825
Base case	70750
-10%	63675
-50%	35375

As mentioned in the introduction (section 1.4) during overcoking it is expected that the SiC content increases, and that the concentration of SiO gas decreases in the low temperature zone. It is however not expected that the Si production increases immediately as it takes time for the increase in SiC(s) concentration to reach the entire system. The effects on the temperature profiles will be interesting. The reaction between C(s)

and SiO(g) is exothermic, leading to an temperature increase, and possibly also less condensation of SiO(g), which again decreases the reaction between liquid quartz and carbon.

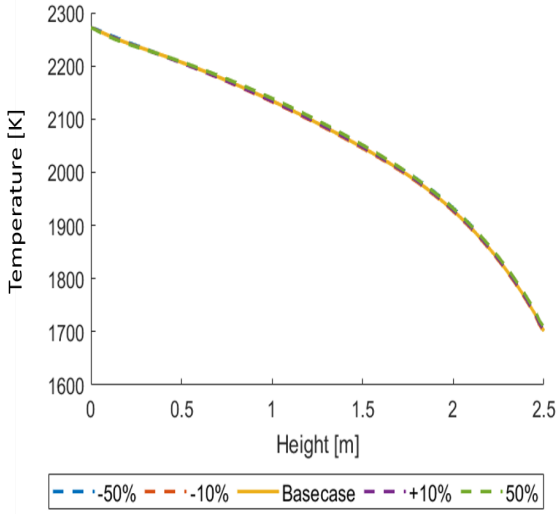
If the low temperature zone is undercoked it is expected that the concentration of SiC decreases in the system. As the concentration of SiC(s) in the low temperature zone changes, in a real furnace, the SiC(s) concentration in the high temperature zone would also change accordingly, affecting the concentration of the gas species SiO(g) and CO(g) leaving the high temperature zone and entering the low temperature zone. This is not captured here at the composition of the gas entering the low temperature zone is kept constant. As there is lower amount of C to react with SiO gas, it is expected that the condensation rate of SiO gas to increase. The temperature profile is also expected to change accordingly, as they have different reaction enthalpies even though they are both exothermic reactions. However, a higher production rate of liquid quartz due to condensation of SiO gas might lead to more liquid quartz reacting with solid carbon, which is an strongly endothermic reaction, causing the gas temperature to decrease.

In figure 7.7a the temperature profile of the gas phase has been plotted with different concentrations of carbon entering the low temperature zone. In table 7.3 the values of the gas temperature leaving the furnace at $z = 2.5$ m are shown for the different carbon concentrations. It can be observed that the gas temperature did not change drastically, but higher carbon concentrations yield higher gas temperatures leaving the low temperature model. However, compared to the substantial effect of the change in reactivity the change in carbon concentration gives less change. In figure 7.7b the graph shows the SiC-yield with different concentrations of carbon. A higher carbon concentration gives a higher SiC-yield. The SiC-yield increases from the top to 0.2 m from the bottom. The reduction below 0.2 m is due to the silicon production reaction R_4.

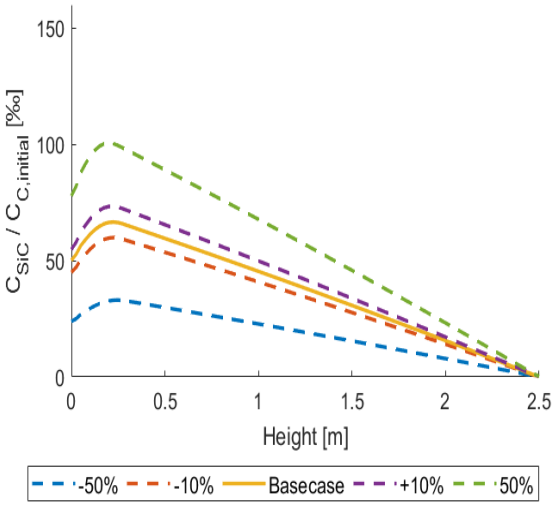
In figure 7.8 shows how the change in concentration input to the low temperature model for three different times to illustrate the functionality of the framework implemented in OpenFOAM.

Table 7.3: Temperature of the gas at the outlet ($z = 2.5$ m) when system has stabilized with different concentrations of carbon

Change in carbon concentration	$\mathbf{T}_{g,out}$**[K]**
-50%	1703.85
-10%	1700.12
Base case	1701.66
+10%	1703.04
+50%	1708.37

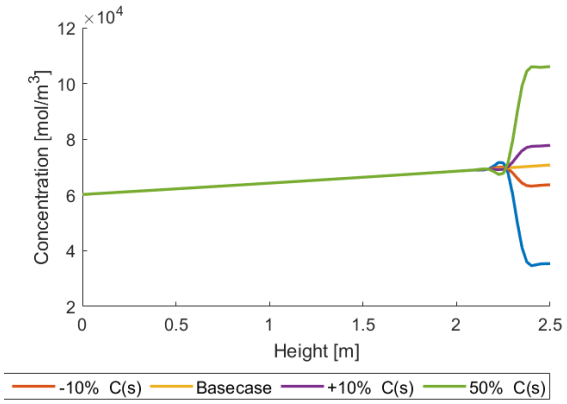


(a) Gas temperature profile with different concentrations of carbon

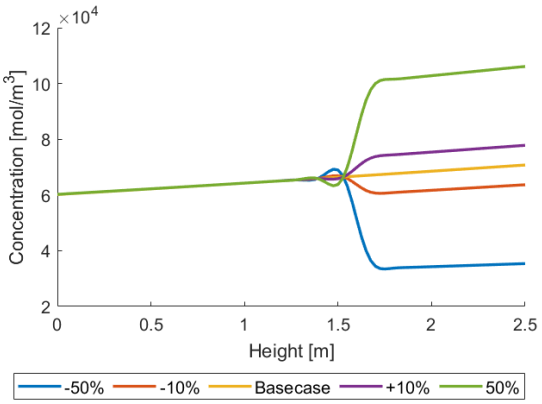


(b) SiC yield with different concentrations of carbon

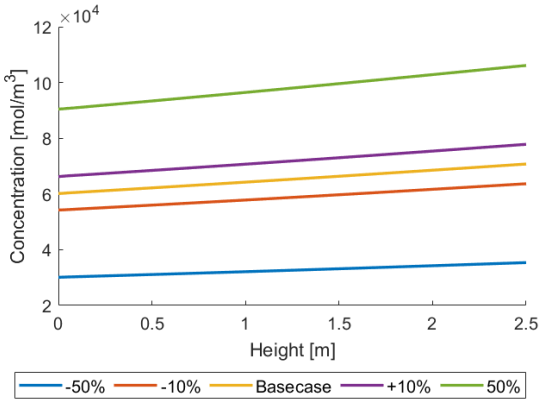
Figure 7.7: Temperature profiles of the gas phase for four different values of carbon reactivity



(a) Carbon concentration after $t = 20$ min.



(b) Carbon concentration after $t = 100$ min



(c) Carbon concentration after $t = 13400$ min.

Figure 7.8: Response to change in concentration input to the low temperature model, shown at three different times to illustrate the functionality of OpenFOAM

7.4 High temperature: Change in inlet composition

As discussed in the introduction the charge composition can be changed to either change the final metal composition or secure that the oven runs well, which also affects the composition in the high temperature area. This case will therefore investigate how the change in composition of charge material affects the concentration and temperature profiles in the high temperature area described by our framework. The change in carbon material affects the concentration of SiC entering the high temperature area, which then again affect the silicon yield, temperature profiles and concentration profiles. In this case we keep the concentration of liquid quartz to zero and $r_C = 0.1$. We will investigate the affect of different yields of SiC(s) coming out of the low temperature zone. In the base case, we assume that 70% of the carbon has reacted to SiC(s). We calculate the concentration of SiC(s) based on the conversion of C(s) entering the high temperature zone. We set the conversion rates to be 50%, 60%, 70%, 80%, and 90%. We evaluate the steady-state change of the temperature profile of the gas phase, and the change in silicon yield. Silicon yield is calculated in the same manner as SiC yield:

$$\text{Si Yield (thousandth)} = \frac{\text{Total Si content at each position}}{\text{Total quartz}} \times 1000, \tag{7.2}$$

In order to ensure stabilization, we simulate the system (after changing the SiC concentration) for 50% longer than required for all the charge to have been replaced. This means we first run the case with 70% SiC conversion for 32400 s, then change the inlet composition of SiC and let the four different cases run for another 16000 s. The calculations done when changing the inlet concentration took around 77 s in execution time for each case.

When the low temperature zone is undercoked we expect that there is not so much SiC(s) available as in the base case in the high temperature area. With lower concentrations of SiC(s) we would expect that there is less production of liquid silicon and CO gas according to reaction R_4: SiO(g) + SiC(s) $\longrightarrow$ 2 Si (l) + CO(g). However, as there is a higher concentration of C(s), it would be expected that there would be higher production of SiO gas according to reaction R_5: SiO_2(l) + C(s) $\longrightarrow$ CO(g) + SiO(g). Because it is expected that the reaction rate of reaction R_5 increases and reaction rate of reaction R_4 decreases, we would expect an temperature decrease when the system is undercoked in comparison with the base case (as reaction R_5 is more endothermic than reaction R_4). For the overcoked case we would expect higher concentrations of SiC(s), so higher reaction rate for reaction R_4, and hence higher temperatures than in the base case, in addition to higher concentrations of Si(l) and CO(g). Less concentration of carbon would cause less production of CO(g) and SiO(g) according to reaction R_5.

Figure 7.9a shows the temperature of the gas in the high temperature model. The model shows very little change as a response to changes in SiC concentrations, this is also shown in table 7.4.

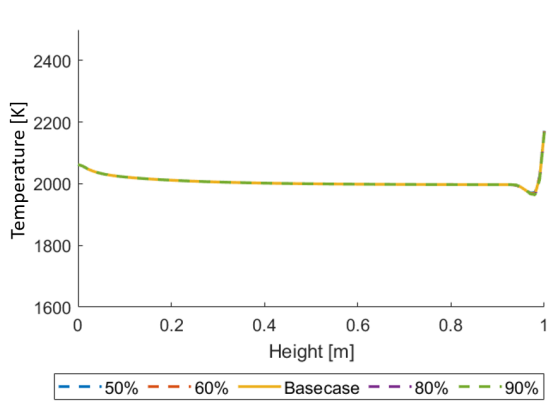
The plot in figure 7.9b shows the reaction rate for the reaction R_4, which indicates that the silicon production takes place very near the bottom of the furnace in the high temperature zone. The model shows very little difference between the different SiC concentrations. Figure 7.9c and table 7.5 show the silicon yield, which varies from 0 to 75‰in the domain, but varies little due to change in SiC(s) concentration. The change in SiC(s) concentration is illustrated in figure 7.10.

Table 7.4: Temperature of the gas at the outlet ($z = 2.5m$) when system has stabilized with different conversion of carbon

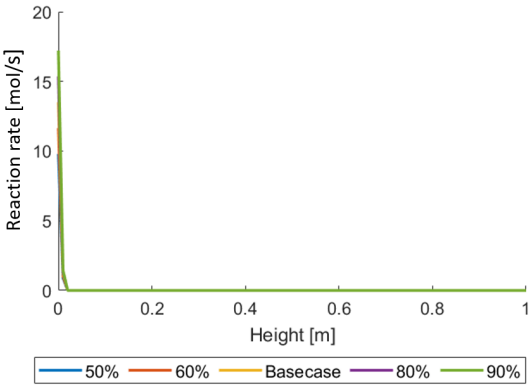
Conversion of C(s)	**T$_{g,out}$[K]**
50%	2062.53
60%	2062.47
basecase	2062.41
80%	2062.37
90%	2062.31

Table 7.5: Silicon yield at $x = 0$ m when the system has stabilized with different compositions of SiC entering the high temperature model.

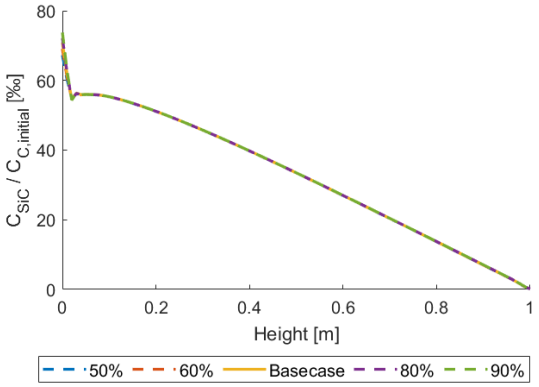
Value of C(s) conversion	**Si Yield [‰]**
50%	0.1355
60%	0.1389
basecase	0.1422
80%	0.1454
90%	0.1484



(a) Temperature of the gas phase

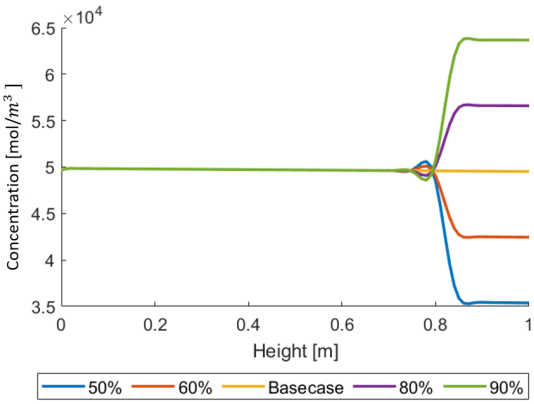


(b) Reaction rate R_4 when the system has stabilized

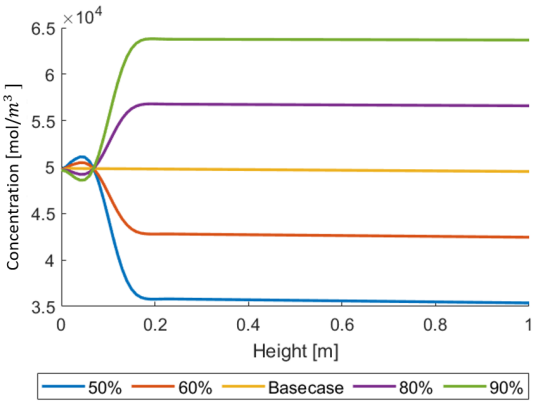


(c) Si(l) yield

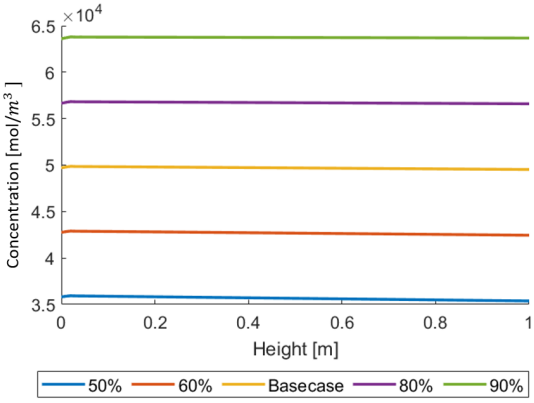
Figure 7.9: Results from change in SiC(s) concentration entering the high temperature zone.



(a) SiC concentration after t=20 min.



(b) SiC concentration after t=100 min



(c) SiC concentration after t=13400 min.

Figure 7.10: Response to change in concentration input to the high temperature model, shown at three different times to illustrate the functionality of OpenFOAM

Chapter 8

Discussion

A verification study was done in order to verify some aspects of the framework. The mass balance for each species in each reaction was conserved. The verification study of the heat and mass balance of the framework was however done for an older version of the framework, and should therefore be done for the present framework in further work. Calculations of electrode heat generation were verified by comparing to an analytical solution, which yielded a satisfactory result. However, a verification of the new developed electrode heat generation should be done. One interesting result from the verification work was the velocity of the temperature, or movement of heat. It was observed that the gas temperature moved with the intrinsic velocity, and not the superficial velocity with the implemented equation. It was assumed that the velocities in this model are constants, that also includes the molar velocities. This is a simplification that should be investigated further, as there are chemical reactions in the system, and assuming that the solid and gas densities are constant is a simplifying assumption. With constant molar velocities it is not possible to capture change in the molar velocities due to chemical reactions in the present work, and this should be further developed in future work.

The heat transfer coefficient calculated using the method presented in Nield and Bejan [18], yielded a temperature profile for the gas and solid phase where the temperature of the solid phase reaches the temperature of the gas phase when the system had stabilized. It is however not expected that the temperature of the solid phase in the low temperature area reaches the gas temperature when the system has stabilized, as the temperature of the gas is quite high (see figure 6.27). The reason for this mismatch may be that the gas velocity in the void space is a lot higher than in the solid pores, which yields a greater heat transfer coefficient, as the gas in a silicon furnace often passes though in channels. This should however be further looked into in future work and a temperature dependent heat transfer coefficient should be implemented. One idea might be to multiply with an adjusting factor to decrease the size of the heat transfer coefficient, while keeping it temperature dependent. For a more realistic heat transfer, a constant heat transfer coefficient was used for obtaining the results in chapter 7.

The calculated off-gas temperature for the base case of the low temperature model yielded a temperature of 1687 K (1414 °C), which is not far off from the measured temperature of 1400°C according to Schei et al. [1]. However, grid convergence for the framework was not investigated. According to Versteeg and Malalasekera [29] only grid converged results can be trusted as otherwise the results may be inaccurate. Grid convergence should therefore be evaluated in further work on the model. Still, the results indicate that the model is capable of calculating reasonable temperatures with this framework and the simplifications done in the model. On the other hand this might also be due to the many errors such as low reaction rate constants. Some of the main assumptions were that I assumed no heat transfer due to radiation and conduction. This

should be implemented in further work as it can affect the temperature calculations.

The effects of changing the carbon reactivity in the low temperature model was investigated. As shown in figures 7.1-7.2b, the temperature and reaction rate profiles are reasonable. From figure 7.3b, the reaction rate R_5 (reaction (3.5)) vary between 0.005-0.01 mol/m^{3}s, showing low reaction rates in accordance with Schei et al. [1].

With increasing carbon reactivity the SiC-yield increases, consequently increasing reaction rate R_4 due to more formation of SiC(s) as expected. The gas temperature increased with increasing carbon reactivity, which was expected as the reaction between SiO(g) and C(s) is exothermic. On the other hand the SiC-yield around $25 - 80‰$is far too low compared to what is suggested by Schei et al. [1]. The reason for that is likely the low reaction constant that was chosen. The model did not function with a higher reaction rate, the reason for this was not found due to time constraints. The figure 7.5a indicates reasonable prediction of the melting of quartz when it comes to the z-position comparing with the results from Nordnes [43].

The plot in figure 7.4b shows the concentration of the SiO-gas in the low temperature zone. This plot shows the weaknesses in the framework, the variation of the concentration seems to vary too little with the height. The model can capture the effect of variation in carbon reactivity as with higher carbon reactivity the concentration of SiO(g) decreases in the domain. In figure 7.4a the plot shows the concentration of the liquid silicon with different carbon reactivities. The flat curve from 0.2 to 1.7 m is due the condensation reaction, and the high concentration from 0 to 0.2 m is due to the Si production. A higher carbon reactivity gives a higher concentration of Si(l) below 0.2 m, which makes sense as a higher carbon reactivity will yield higher SiC(s) concentrations and higher reaction rates R_4. However, between 0.2-1.7 m, the concentration of Si(l) decreases with increasing carbon reactivity. This is because higher carbon reactivity increases the gas temperature (as seen in figure 7.1), which gives a lower reaction rate for the condensation of SiO(g), R_2 (as shown in figure 7.3a). This is a result of increased $P_{SiO,equ,2}$ due to increasing temperature (shown in figure 3.3), hence R_2 decreases with increasing gas temperature yielding lower concentrations of Si(l) with increasing carbon reactivity.

The effects of change in charge composition in the low temperature model was investigated. Figure 7.7a shows a qualitative reasonable temperature profile, but no change in temperature due to change in charge composition. In table 7.3 the gas temperature leaving the furnace at $z = 2.5$ m are presented, showing little change in gas temperature due to change in carbon concentration. The figure 7.7b shows again too low SiC-yield, even though the model could capture the change in SiC-yield due to different carbon concentration. This might again be due to the low reaction rates used in the framework, but does indicate the the reactivity of carbon has an higher effect on the temperature and concentration profiles than the concentration of carbon in the present framework. When changing the concentration of carbon the molar concentrations should have changed as well, but these were kept constant and is not captured in this model in which might have affected the results. The simulation time of 13400 min was chosen to secure the steady state of the raw material transport as shown in figure 7.8.

The response of the high temperature model to changing concentrations was also evaluated, by changing the concentration of SiC(s) and C(s) entering the high temperature model. Figure 7.9 shows temperature, reaction rate and Si-yield as a function of different SiC-concentrations. The temperature seems not to be affected by the change in SiC concentration. Both the figure 7.9b and 7.9c shows results that is not likely to be correct as production of Si(l) is expected to occur in the entire domain of the high temperature model. The Si-yield was calculated to be between 0 - 75‰mol/m^3. The yield is an order of magnitude off from an industrial process, where the yield is around 80-90% [1]. Also these results may not correspond to the industrial process due to the weakness of the tuning of the framework. The reason for this huge deviation seems to be that the necessary energy is not provided in the model, in addition to low reaction rates as discussed earlier. One error is also that the concentration of liquid quartz was set to zero. This should be

changed in future work, as the materials in the real process drips from the low temperature zone down to the high temperature zone [1].

OpenFOAM seems to be useful to make a comprehensive framework of the silicon process. This work also shows examples of information and parameters that may be calculated and presented from the process. Some necessary simplifications has to be discussed and corrected in a future model. This model did not succeed in direct connection between high and low temperature zone, but proves that information can be taken out from one zone and used in the other in future work. Coupling of the high and low temperature models will give a longer computing time. However with proper tuning OpenFOAM seems to be able to give fast models. The man-hours spent to create a fast model and to tune the model gave very little time for the optimization of the process. The results in chapter 6 are in some ways not representative for the real silicon process.

Chapter 9

Conclusion

In this work, a framework for models of the low and high temperature zone of the silicon process was developed in OpenFOAM. OpenFOAM seemed to be useful to make a comprehensive framework of the silicon process. Some essential features of the framework were verified, such as heat and mass balance and electrode heat generation. The framework however need further development, some necessary simplifications has to be discussed and corrected in a future model. The results show that information can be taken out of each sub-model and used in the other in order to couple the two models dynamically in future work.

The work in this thesis shows that OpenFOAM has potential to be used for dynamic simulations of the silicon production process, and can be explored further.

9.1 Further work

The main further work of this model are presented below:

- Implement a temperature dependent heat transfer coefficient.
- Implement higher reaction rates
- Investigate mesh convergence.
- Implement non constant velocities
- Extend the model to 2-and 3D in order to validate the model
- Implement heat transfer due to conduction and radiation
- Crete several sub-models which can communicate with each other and describe the different parts of a furnace rapidly, in order to obtain gas temperature leaving the furnace.
- Implement carbon reactivity as a changeable variable

List of Notation

Symbols

s	:	When used after a chemical species, denotes that the species is in its solid state	[]
l	:	When used after a chemical species, denotes that the species is in its liquid state	[]
g	:	When used after a chemical species, denotes that the species is in its gaseous state	[]
h	:	Height of model	[m]
x,y,z	:	Coordinates	[m]
T_g	:	Temperature of gas phase	[K]
T_s	:	Temperature of joined solid and liquid phase	[K]
R_j	:	Reaction rate for reaction j	[mol/sm^3]
C_i	:	Concentration of species i	[mol/m^3]
φ	:	Porosity	[]
$v_f^{'}$	:	Intrinsic fluid velocity	[m/s]
v_f	:	Superficial fluid velocity	[m/s]
V_m	:	Total volume of medium, including solid and fluid material	[m^3]
ρ_g	:	Density of gas phase	[kg/m^3]
ρ_s	:	Density of joined solid-liquid phase	[kg/m^3]
$\vec{v}$	:	Superficial velocity vector of gas phase	[m/s]
$\dot{R}$	:	Source term	[kg/sm^3]
U_g	:	Superficial gas velocity in z direction	[m/s]
U_s	:	Superficial solid (joined solid and liquid) velocity in z direction	[m/s]
$U_{g,m}$	:	Superficial molar velocity of gas phase in z direction	[m/s]
$U_{s,m}$	:	Superficial molar velocity of solid phase (joined solid and liquid) in z direction	[m/s]
$\dot{j}_i$	:	Mass flux species i	[kg/sm^2]
$j_{m,i}$	:	Molar flux species i	[mol/sm^2]
M_i	:	Molar mass species i	[g/mol]
C_g	:	Total concentration of in the gas phase	[mol/m^3]
C_s	:	Total concentration of species in the joined solid-liquid phase	[mol/m^3]
k	:	Permeability of charge	[m^2]
Δp	:	Pressure gradient due to pressure drop along the z axis	[atm]
μ	:	Dynamic viscosity of the gas	[kg/ms]
P_{TOT}	:	Total pressure	[atm]
P_i	:	Partial pressure of species i	[atm]
R	:	Gas constant	[kg/m^2]
E_j	:	Activation energy for reaction j	[kg/ms^2]
r_C	:	Reactivity of carbon	[]

$P_{SiO,equ}$	:	Equilibrium partial pressure of reaction j	[kg/ms^2]
ΔG_{Rj}	:	Change in Gibbs free energy	[J/mol]
K_{equ}	:	Equilibrium chemical reaction constant	[]
k_j	:	Rection rate constants used for reaction j	various
k_+	:	Forward rate constant	various
k_-	:	Backward rate constant	various
a_j	:	Constant used to calculate ΔG_{R_j}	[J/mol]
b_j	:	Constant used to calculate ΔG_{R_j}	[J/molK]
d_j	:	Constant used to calculate ΔG_{R_j}	[J/molK]
c	:	Specific heat of solid phase	[J/kgK]
c_p	:	Specific heat at constant pressure of gas phase	[J/kgK]
k_s	:	Thermal conductivity of solid phase	[]
k_g	:	Thermal conductivity of gas phase	[]
h	:	Heat transfer coefficient	[W/m^{3}K]
h^*	:	Heat transfer coefficient	[W/m^{2}K]
q_g'''	:	Heat production per unit volume in gas phase	[W/m^3]
q_s'''	:	Heat production per unit volume in joined solid-liquid phase	[W/m^3]
$\Delta H_{j,s}$	:	Reaction enthalpy contribution to solid phase due to chemical reaction	[J/mol]
$\Delta_f H_i^\circ$	:	standard enthalpy of formation of species j	[J/mol]
$\Delta_f H_i(T)$	:	Reaction enthalpy of formation of species i at temperature T	[J/mol]
a_{sg}	:	Specific surface area	[1/m]
d_p	:	Particle diameter	[m]
Re_p	:	Reynolds number based on particle diameter	[]
Nu_{gs}	:	Gas-to-solid Nusselt number	[]
Q_{el}	:	Heat production due to electrode heat generation	[W/m^3]
u,v	:	Velocity components in x and y direction	[m/s]
t	:	Time	[s]
$\dot{m}_{SiO_2}$	:	Mass flow rate of quartz	[kg/g]
$\dot{m}_C$	:	Mass flow rate of C	[kg/g]
$\dot{m}_{SiO}$	:	Rate of SiO production	[kg/h]
$\dot{m}_{CO}$	:	Rate of CO(g) production	[kg/h]
$d_{internal}$	:	Internal diameter of electrode	[m]
H_{charge}	:	Height of charge	[m]
N_e	:	Number of electrodes	[]
ρ_{bulk}	:	Density of bulk phase	[kg/m^3]
$\dot{V}$	:	Volumetric flow rate	[m^3/s]
$A_{ac,tot}$	:	Total active area around the three electrodes	[m^2]
$A_{ac,el}$	:	Active area around each electrode	[m^2]
R_{ac}	:	Radius around each electrode	[m]
W_i	:	Weight fraction of species i	[]
x_i	:	Mole fraction of species i	[]
n	:	Number of moles	[mol]
V	:	Volume	[]
ν	:	Kinematic viscosity	[m^2/s]
ε_C	:	Porosity of carbon	[]

Sub- and superscripts

j	=	Chemical reaction
i	=	Chemical specie
f	=	Arbitrary face
e	=	East face
n	=	North face
s	=	South face
w	=	West face
ac	=	Active area

Abbreviations

LT model	=	Low temperature model
HT model	=	High temperature model

Bibliography

[1] Anders Schei, J. Tuset, and H. Tveit. *Production of high silicon alloys*. Tapir, Trondheim, 1998.

[2] Halvard Tveit and Leiv Kolbeinsen. The (love and hate) role of entropy in process metallurgy. *13th International Conference on Society & Materials, SAM13, Pisa*, pages 20–21, May 2019.

[3] Mehdi Kadkhodabeigi. *Modeling of Tapping Processes in Submerged Arc Furnaces*. PhD thesis, The Norwegianuniversity of Science and Technology, Trondheim, 2011.

[4] Aasgeir Mikael Valderhaug. *Modelling and Control of Submerged-Arc Ferrosilicon Furnaces*. PhD thesis, The Norwegian Institute of Technology, Trondheim, 1992.

[5] Benjamin Sloman, Colin Please, Robert Van Gorder, Aasgeir Valderhaug, Rolf Birkeland, and Harald Wegge. A heat and mass transfer model of a silicon pilot furnace. *Metallurgical and Materials Transactions B*, 48(5):2664–2676, 2017.

[6] Merete Tangstad. *Metal production in Norway*. Akademika Publishing, Oslo, 2013.

[7] A. Valderhaug and .P Sletfjerding. A non-interacting electrode current controller for submerged-arc furnaces. *Electric furnace conference proceedings*, 49:311–20, 1991.

[8] M. Takla, N.E. Kamfjord, Halvard Tveit, and S. Kjelstrup. Energy and exergy analysis of the silicon production process. *Energy*, 58:138–146, September 2013.

[9] Topic: Silicon, 2017. Retrieved from: https://www.statista.com/topics/1959/silicon/, 2017.

[10] A. Valderhaug, J.G. Balchen, and S.A. Halvorsen. Modelbased control of the ferrosilicon process. *IFAC Proceedings Volumes*, 25(5):117–123, 1992.

[11] S.T Johansen, S Graadahl, R Gammelsaeter, M Raanes, A E Arntsberg, T Lindstad, G Enstad, and H Tveit. Clogging of ferro-silicon furnace off-gas channels at high temperatures, Jan 1992.

[12] J.K Tuset and O. Raaness. Reactivity of reduction materials for the production of silicon, silicon-rich ferroalloys and silicon carbide. *Electric Furnace Conf.*, 34:101–107, 1976.

[13] A. Schei and S. Halvorsen. A stoichiometric model of the silicon process. *Proceedings from: The Ketil Motsfeldt symposium*, pages 41–56, 1991.

[14] S. Halvorsen, A Schei, and J. Downing. A unidimensional dynamic model for the (ferro)silicon process. *Electr.Furn.Conf.Proc.*, 50:45–59, 1992.

[15] Svenn A. Halvorsen. A UNIDIMENSIONAL DYNAMIC MODEL FOR THE FERROSILICON PROCESS - A RELIABLE TOOL FOR IDENTIFYING THE STATE OF THE FURNACE. *Proceedings of the Conference Inverse Problems and Optimal Design in Indus*, pages 229–238, 1994.

[16] Birger Andresen. *Process model for carbothermic production of silicon metal*. PhD thesis, The Norwegian Institute of Technology, Trondheim, 2012.

[17] Kjell Wiik. *Kinetics of reactions between silica and carbon*. PhD thesis, Institutt for uorganisk kjemi, Norges tekniske høgskole, Trondheim, 1990.

[18] Donald A Nield and Adrian Bejan. *Convection in porous media*. Springer, 3rd ed. edition, 2006.

[19] Halvard Tveit. Private Communication, fictional furnace calculations, 2018.

[20] W. B. Fulks, R. B. Guenther, and E. L. Roetman. Equations of motion and continuity for fluid flow in a porous medium. *Acta Mechanica*, 12(1):121–129, 1971-03.

[21] D.E. Rosner. *Transport processes in chemically reacting flow systems*. Butterworth Publishers,Stoneham, MA, 1986.

[22] Hugo A. Jakobsen. *Chemical Reactor Modeling: Multiphase Reactive Flows*, volume 9783319050928. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[23] Kaplan S. Basniev, Nikolay M. Dmitriev, and George V. Chilingar. *Mechanics of Fluid Flow*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2012.

[24] David R Gaskell. Introduction to metallurgical thermodynamics, 1981.

[25] Anthony G. Dixon and David L. Cresswell. Theoretical prediction of effective heat transfer parameters in packed beds. *AIChE Journal*, 25(4):663–676, 1979.

[26] H. Miyauchi, H. Kataoka, and T. Kikuchi. Gas film coefficient of mass transfer in low pclet number region for sphere packed beds: Chem. engng sci. 1976 31 9., 1976.

[27] N. Wakao, K. Tanaka, and H. Nagai. Measurements of particle-to-gas mass transfer coefficients from chromatographic adsorption experiments. *Chemical Engineering Science*, 31(12):1109–1113, 1976.

[28] N. Wakao, S. Kaguei, and T. Funazkri. Effect of fluid dispersion coefficients on particle-to-fluid heat transfer coefficients in packed beds: Correlation of nusselt numbers. *Chemical Engineering Science*, 34(3):325–336, 1979.

[29] H.K Versteeg. An introduction to computational fluid dynamics : the finite volume method, 2007.

[30] Sindre E. Gylver. Alumina dissolution in cryolite melts. Master's thesis, Norwegian University of Science and Technology(NTNU), Trondheim, 2018.

[31] The OpenFOAM Foundation. Openfoam and the openfoam foundation. https://openfoam.org/. (accessed: 01.02.2019).

[32] OpenCFD (2019). laplacianfoam file reference. https://www.openfoam.com/documentation/guides/latest/doc/guide-applications-solvers-basic-laplacianFoam.html. (accessed: 01.02.2019).

[33] Fatbardha Hajdini. Metallurgical process modelling for sustainability, specialization project. 2018.

[34] Permeability - an overview | ScienceDirect topics. https://www.sciencedirect.com/topics/materials-science/permeability. Accessed 2019-05-27.

[35] HSC 9.1 and HSC Sim,QUOTEC: *Pure Substance Database, Various Standard Application Modules, Process Simulation Linking Minerals and Materials Processing, Envirommental Assesment,* http://www.outotec.com/hsc linked to the REA - REACTION EQUATIONS module http://www.outotec.com/products/digital-solutions/hsc-chemistry/hsc-rea—reaction-equations-module/ and and Thermochemical Database http://www.outotec.com/products/digital-solutions/hsc-chemistry/hsc-thermochemical/, Accessed Mai 2019.

[36] Allan Blackman. Aylward and Findlay's SI Chemical Data, 2014.

[37] Yunus A Çengel, , and John M. Cimbala. *Fluid mechanics : fundamentals and applications*. McGraw-Hill, Boston, 3rd ed. in si units. edition, 2014.

[38] Yunus A engel. *Fluid mechanics : fundamentals and applications*. McGraw-Hill, Boston, 3rd ed. in si units. edition, 2014.

[39] Akito Kasai, Takeaki Murayama, and Yoichi Ono. Measurement of effective thermal conductivity of coke. *ISIJ International*, 33(6):697–702, 1993.

[40] F. F. Lezhenin and G. G. Gnesin. Thermal conductivity of silicon carbide at high temperatures. *Soviet Powder Metallurgy and Metal Ceramics*, 6(2):114–116, 1967-02.

[41] Michal Ksiazek, T Manik, M Tangstad, and E Ringdalen. The thermal diffusivity of raw materials for ferromanganese production. page 9.

[42] H.R. Shanks, P.D. Maycock, P.H. Sidles, and G.C. Danielson. Thermal conductivity of silicon from 300 to 1400k. *Physical Review*, 130(5):1743–1748, 1963.

[43] Elisabeth Nordnes. Softening and melting properties of quartz. Master's thesis, Norwegian University of Science and Technology(NTNU), Trondheim, 2019.

Appendix

In this appendix the code for the low temperature model is presented, as well as changes needed to be done in order to obtain the high temperature model. The models have been sent to my supervisor Kristian Etienne Einardsrud and can be obtained by contacting him which is advised if you want to test the models. In section 4.6 the file structure of the solver and base case has been described.

Appendix A

Low Temperature Model

In this appendix the implementation code for the base case of the low temperature model is presented.

A.1 Solver

The solver created is presented in this section.

A.1.1 ergunMasterFoam.C

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Copyright (C) 2011-2018 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.


\*---------------------------------------------------------------------------*/

#include "fvCFD.H"
#include "fvOptions.H"  //30.10
#include "simpleControl.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{
    #include "setRootCaseLists.H"
    #include "createTime.H"
    #include "createMesh.H"

    simpleControl simple(mesh);

    #include "createFields.H"
    #include "pressureCalc.H"
    #include "viscosity.H"
    #include "themCond.H"

    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
   // Info<< "\nCalculating preassure distribution\n" << endl;

    while (simple.loop(runTime))
    {
        Info<< "Time = " << runTime.timeName() << nl << endl;

 while (simple.correctNonOrthogonal())
        {
           solve
   (
fvm::laplacian(k/mu, p)
   );
        }


#include "viscosity.H" //Need to calculate viscosity before velocity for ergun!!!!


Ug=-k/mu*fvc::grad(p);
Ug.correctBoundaryConditions();


phig = linearInterpolate(Ug) & mesh.Sf(); //The surface flux phi is updated from the new value of the velocity profile Ug
phis = linearInterpolate(Us) & mesh.Sf(); //The surface flux phi is updated from the new value of the velocity profile Us


phigm = linearInterpolate(Ug_m) & mesh.Sf(); //The surface flux phi is updated from the new value of the velocity profile Ug_m
phism = linearInterpolate(Us_m) & mesh.Sf(); //The surface flux phi is updated from the new value of the velocity profile Us_m



#include "reactionRates.H"
#include "reactionHeatGas.H"
//#include "elHeatSource.H"
#include "themCond.H"
//#include "heatExchangeCoeffCalc.H"

fvScalarMatrix TgEqn
(
eps*rho_g*Cpspecific_g*fvm::ddt(Tg) +rho_g*Cpspecific_g*fvm::div(phig,Tg)
==
eps*qg-fvm::Sp(h_0,Tg)+h_0*Ts
);
TgEqn.solve();

fvScalarMatrix TsEqn
(
(1.-eps)*rho_s*Cpspecific_s*fvm::ddt(Ts) + rho_s*Cpspecific_s*fvm::div(phis,Ts)
==
(1-eps)*qs-fvm::Sp(h_0,Ts)+h_0*Tg
);
TsEqn.solve();
#include "massBalance.H"
#include "pressureCalc.H"

runTime.write();

        Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
            << "  ClockTime = " << runTime.elapsedClockTime() << " s"
            << nl << endl;
    }

    Info<< "End\n" << endl;

    return 0;
}

// ************************************************************************* //
```

A.1.2 elHeatSource.H

```
//Info<< "Solver: elHeatSource.H \n" << endl;

volScalarField Q_el_a
(
    IOobject
    (
        "Q_el_a",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::NO_WRITE
    ),
    mesh,
    dimensionedScalar("Q_el_a",dimensionSet(1,-1,-3,0,0,0,0),200) //[Q_el]= W/m=kg/sm;
```

```
);


volScalarField Q_el_b
(
    IOobject
    (
        "Q_el_b",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::NO_WRITE
    ),
    mesh,
    dimensionedScalar("Q_el_b",dimensionSet(1,-1,-3,0,0,0,0),5) //[Q_el]= W/m=kg/sm;

);

volScalarField oneEl
(
    IOobject
    (
        "oneEl",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,              // When using NO_READ we should initiliaze our variable, but I want it dependent og C_SiO and C_CO
        IOobject::NO_WRITE
    ),
    mesh,
    dimensionedScalar("oneEl",dimensionSet(1,-1,-3,0,0,0,0),1.0)
);

volVectorField centres = Us.mesh().C();
volScalarField x = centres.component(0);
          Info<< min(x) << nl << endl;
          Info<< max(x) << nl << endl;

Q_el=Q_el_a*(-1+exp((Q_el_b/oneEl)*(1-x/L)));
```

A.1.3 heatexchangeCoeffCalc.H

```
//This file is used to calculate the old heat exchange coefficient, and is deactivated, but the code is presented here:
volScalarField Nu_fs
(
    IOobject
    (
        "Nu_fs",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("Nu_fs",dimensionSet(0,0,0,0,0,0,0),0.0)
);

volScalarField ledd1
(
    IOobject
    (
        "ledd1",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("ledd1",dimensionSet(-1,0,3,1,0,0,0),0.0)
);

volScalarField ledd2
(
    IOobject
    (
        "ledd2",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("ledd2",dimensionSet(-1,0,3,1,0,0,0),0.0)
);

volScalarField Re_p
(
    IOobject
```

```
    (
        "Re_p",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("Re_p",dimensionSet(-1,3,0,0,0,0,0),0.0)
);

volScalarField Re_p_dim
(
    IOobject
    (
        "Re_p_dim",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("Re_p_dim",dimensionSet(-1,3,0,0,0,0,0),1)
);

Re_p=(dp/my)*mag(Ug);

volScalarField Re_p_test=Re_p/Re_p_dim;

Nu_fs=(0.255/eps)*pow(Pr,0.333)*pow(Re_p_test,0.667);

ledd1=(dp/(Nu_fs*kg_T));
ledd2=dp/(beta*ks_T);

volScalarField h_star_under= (dp/(Nu_fs*kg_T))+dp/(beta*ks_T);
h_star=1/h_star_under;
dimensionedScalar a_fs=6*(1-eps)/dp;
h=a_fs*h_star;
```

A.1.4 massBalance

```
//Mass balance calculations are done in this file
/*
All species we wish to solve for:
C
SiC
SiO2_s
SiO2_l
Si
CO
SiO
*/

//All the mass balances:
fvScalarMatrix CEqn
(
(1-eps)*fvm::ddt(C_C) + fvm::div(phism,C_C)
==
(1-eps)*(-2*R_1-R_5)

);
CEqn.solve();

fvScalarMatrix SiCEqn
(
(1-eps)*fvm::ddt(C_SiC) + fvm::div(phism,C_SiC)
==
(1-eps)*(R_1-R_4)

);
SiCEqn.solve();


fvScalarMatrix SiO2_sEqn
(
(1-eps)*fvm::ddt(C_SiO2_s) + fvm::div(phism,C_SiO2_s)
==
(1-eps)*(-R_3)

);
SiO2_sEqn.solve();


fvScalarMatrix SiO2_lEqn
(
(1-eps)*fvm::ddt(C_SiO2_l) + fvm::div(phism,C_SiO2_l)
==
```

```
(1-eps)*(R_2-R_neg2+R_3-R_5)

);
SiO2_lEqn.solve();


fvScalarMatrix SiEqn
(
(1-eps)*fvm::ddt(C_Si) + fvm::div(phism,C_Si)
==
(1-eps)*(R_2-R_neg2+2*R_4)

);
SiEqn.solve();



//MassBalance for the gas phase

fvScalarMatrix COEqn
(
(eps)*fvm::ddt(C_CO) + fvm::div(phigm,C_CO)
==
(eps)*(R_1 + R_4 + R_5)

);
COEqn.solve();


fvScalarMatrix SiOEqn
(
(eps)*fvm::ddt(C_SiO) + fvm::div(phigm,C_SiO)
==
(eps)*(-R_1-2*R_2+2*R_neg2-R_4 + R_5)

);
SiOEqn.solve();
```

A.1.5 pressureCalc.H

```
//Calculation initial total gas concentrations and solid consentrations in addition to partial pressure!

//Info<< "Solver: pressureCalc.H \n" << endl;


volScalarField Pzerro
(
    IOobject
    (
        "Pzerro",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::NO_WRITE
    ),
    mesh,
    dimensionedScalar("Pzerro",dimensionSet(1,-1,-2,0,0,0,0),0.0)
);

volScalarField Czerro
(
    IOobject
    (
        "Czerro",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::NO_WRITE
    ),
    mesh,
    dimensionedScalar("Czerro",dimensionSet(0,-3,0,0,1,0,0),0.0)
);

C_g= max((C_SiO + C_CO), Czerro);
P_SiO= max(((C_SiO/C_g) * P_TOT),Pzerro);
P_CO=max(((C_CO/C_g) * P_TOT),Pzerro);
C_s= max((C_C + C_SiC +C_SiO2_s+ C_SiO2_l + C_Si), Czerro);
```

A.1.6 reactionHeatGas.H

```
//Calculaion of heat production due to reactions and specific heat of the solid and gas phase:
/* A matrix has been made for the enthalpy calculations:
```

```
C c11 c12 c13 c14 c15
Sic c21 c22 c23 c24 c25
SiO2_s c31 c32 c33 c34 c35
SiO2_l c41 c42 c43 c44 c45
Si c51 c52 c53 c54 c55
CO c61 c62 c63 c64 c65
SiO c71 c72 c73 c74 c75

This multiplied with:

T^4
T^3
T^2
T
1

This yiels:
H_i (T) = c_i1T^4 + c_i2T + c_i3T + c_i4T +c_i5, where i is a specie

*/

volScalarField H_1 = c11*pow(Ts,4) + c12*pow(Ts,3) + c13*pow(Ts,2) + c14*pow(Ts,1) +c15; //[J/mol]
volScalarField H_2 = c21*pow(Ts,4) + c22*pow(Ts,3) + c23*pow(Ts,2) + c24*pow(Ts,1) +c25;
volScalarField H_3 = c31*pow(Ts,4) + c32*pow(Ts,3) + c33*pow(Ts,2) + c34*pow(Ts,1) +c35;
volScalarField H_4 = c41*pow(Ts,4) + c42*pow(Ts,3) + c43*pow(Ts,2) + c44*pow(Ts,1) +c45;
volScalarField H_5 = c51*pow(Ts,4) + c52*pow(Ts,3) + c53*pow(Ts,2) + c54*pow(Ts,1) +c55;

volScalarField H_6 = c61*pow(Tg,4) + c62*pow(Tg,3) + c63*pow(Tg,2) + c64*pow(Tg,1) +c65; //g
volScalarField H_7 = c71*pow(Tg,4) + c72*pow(Tg,3) + c73*pow(Tg,2) + c74*pow(Tg,1) +c75; //g

volScalarField H_6_rx = c61*pow(Ts,4) + c62*pow(Ts,3) + c63*pow(Ts,2) + c64*pow(Ts,1) +c65; //g
volScalarField H_7_rx = c71*pow(Ts,4) + c72*pow(Ts,3) + c73*pow(Ts,2) + c74*pow(Ts,1) +c75; //g

volScalarField deltaH_RX1_tot
(
    IOobject
    (
        "deltaH_RX1_tot",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("deltaH_RX1_tot",dimensionSet(1,2,-2,0,-1,0,0),0)

);


volScalarField deltaH_RX2_tot
(
    IOobject
    (
        "deltaH_RX2_tot",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("deltaH_RX2_tot",dimensionSet(1,2,-2,0,-1,0,0),0)

);


volScalarField deltaH_RX3_tot
(
    IOobject
    (
        "deltaH_RX3_tot",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("deltaH_RX3_tot",dimensionSet(1,2,-2,0,-1,0,0),0)

);


volScalarField deltaH_RX4_tot
(
    IOobject
    (
        "deltaH_RX4_tot",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
```

```
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("deltaH_RX4_tot",dimensionSet(1,2,-2,0,-1,0,0),0)

);

volScalarField deltaH_RX5_tot
(
    IOobject
    (
        "deltaH_RX5_tot",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("deltaH_RX5_tot",dimensionSet(1,2,-2,0,-1,0,0),0)
);

volScalarField H_fus_3
(
    IOobject
    (
        "H_fus_3",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("H_fus_3",dimensionSet(1,2,-2,0,-1,0,0),11e6)

);

//Using Tg for gas enthalpies:
deltaH_RX1_tot  = 1*H_2 + 1*H_6_rx  - 1*H_7 - 2*H_1;
deltaH_RX2_tot  = 1*H_4 + 1*H_5 - 2*H_7;
//deltaH_RXNeg22_tot  = 2*H_7-H_4-H_5;
deltaH_RX3_tot  = H_fus_3;
deltaH_RX4_tot  = 2*H_5 + 1* H_6_rx - 1* H_7 - 1* H_2;
deltaH_RX5_tot  = 1*H_7_rx + 1*H_6_rx - 1*H_4 - H_1;

//Reaction enthanlpies for gas phase
volScalarField deltaH_rx1_g   = 1*H_6_rx - 1*H_7; //[J/kmol]
volScalarField deltaH_rx2_g   = -2*H_7;
volScalarField deltaH_rxneg2_g= 2*H_7_rx;
//volScalarField deltaH_rx3_g   = 0;
volScalarField deltaH_rx4_g   = 1* H_6_rx - 1* H_7;
volScalarField deltaH_rx5_g   = 1*H_7_rx + 1*H_6_rx;
volScalarField deltaH_rx2   = 1*H_4 + 1*H_5 - 2*H_7;

//Reaction enthalpies for solid/liquid phase
volScalarField deltaH_rx1_s   = 1*H_2 - 2*H_1; //[J/kmol]
volScalarField deltaH_rx2_s   = 1*H_4 + 1*H_5;
volScalarField deltaH_rxneg2_s= -H_4-H_5;
volScalarField deltaH_rx3_s   =  H_fus_3;
volScalarField deltaH_rx4_s   = 2*H_5 - 1* H_2;
volScalarField deltaH_rx5_s   = -1*H_4 - H_1;

//Calculating heat generation due to reactions in separate phases:

qs= -1*(R_1*deltaH_rx1_s  + R_2*0.9*deltaH_RX2_tot+ R_neg2*deltaH_rxneg2_s + R_3*deltaH_rx3_s
+ R_4*0.7*deltaH_RX4_tot + R_5*0.8*deltaH_RX5_tot);
qg= -1*(R_1*deltaH_rx1_g + R_2*0.1*deltaH_RX2_tot+ R_neg2*deltaH_rxneg2_g + R_4*0.3*deltaH_RX4_tot + R_5*0.2*deltaH_RX5_tot);


//Calculating SPECIFIC heat capacities:
volScalarField Cp_1=a11*pow(Ts,4) + a12*pow(Ts,3) + a13*pow(Ts,2) + a14*pow(Ts,1) +a15;
volScalarField Cp_2=a21*pow(Ts,4) + a22*pow(Ts,3) + a23*pow(Ts,2) + a24*pow(Ts,1) +a25;
volScalarField Cp_3=a31*pow(Ts,4) + a32*pow(Ts,3) + a33*pow(Ts,2) + a34*pow(Ts,1) +a35;
volScalarField Cp_4=a41*pow(Ts,4) + a42*pow(Ts,3) + a43*pow(Ts,2) + a44*pow(Ts,1) +a45;
volScalarField Cp_5=a51*pow(Ts,4) + a52*pow(Ts,3) + a53*pow(Ts,2) + a54*pow(Ts,1) +a55;
//Gasses
volScalarField Cp_6=a61*pow(Tg,4) + a62*pow(Tg,3) + a63*pow(Tg,2) + a64*pow(Tg,1) +a65;
volScalarField Cp_7=a71*pow(Tg,4) + a72*pow(Tg,3) + a73*pow(Tg,2) + a74*pow(Tg,1) +a75;

Cpspecific_s= (C_C/C_s)*(Cp_1/Mm_1) + (C_SiC/C_s)*(Cp_2/Mm_2) +(C_SiO2_s/C_s)*(Cp_3/Mm_3)+
(C_SiO2_l/C_s)*(Cp_4/Mm_4) + (C_Si/C_s)*(Cp_5/Mm_5); //[J/K kg]
Cpspecific_g= (C_CO/C_g)*(Cp_6/Mm_6) + (C_SiO/C_g)*(Cp_7/Mm_7);
```

A.1.7 reactionRates.H

```
volScalarField oneK
(
```

```
    IOobject
    (
        "oneK",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::NO_WRITE
    ),
    mesh,
    dimensionedScalar("oneK",dimensionSet(0,0,0,1,0,0,0),1)
);

volScalarField delta_G_1
(
    IOobject
    (
        "delta_G_1",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("delta_G_1",dimensionSet(1,2,-2,0,-1,0,0),0)
);

volScalarField delta_G_2
(
    IOobject
    (
        "delta_G_2",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,              // When using NO_READ we should initiliaze our variable, but I want it dependent og C_SiO and C_CO
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("delta_G_2",dimensionSet(1,2,-2,0,-1,0,0),0)
);

volScalarField delta_G_4
(
    IOobject
    (
        "delta_G_4",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,              // When using NO_READ we should initiliaze our variable, but I want it dependent og C_SiO and C_CO
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("delta_G_4",dimensionSet(1,2,-2,0,-1,0,0),0)
);

delta_G_1=a1 + b1*Tg*log(Tg/oneK) + c1*Tg;
delta_G_2=a2 + b2*Tg*log(Tg/oneK) + c2*Tg;
delta_G_4=a4 + b4*Tg*log(Tg/oneK) + c4*Tg;

volScalarField one
(
    IOobject
    (
        "one",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,              // When using NO_READ we should initiliaze our variable, but I want it dependent og C_SiO and C_CO
        IOobject::NO_WRITE
    ),
    mesh,
    dimensionedScalar("one",dimensionSet(0,0,0,0,0,0,0),1)
);

volScalarField zerro
(
    IOobject
    (
        "zerro",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,              // When using NO_READ we should initiliaze our variable, but I want it dependent og C_SiO and C_CO
        IOobject::NO_WRITE
    ),
    mesh,
    dimensionedScalar("zerro",dimensionSet(1,-1,-2,0,0,0,0),0.0)
);

volScalarField Tzerro
(
    IOobject
```

```
    (
        "Tzerro",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,              // When using NO_READ we should initiliaze our variable, but I want it dependent og C_SiO and C_CO
        IOobject::NO_WRITE
    ),
    mesh,
    dimensionedScalar("Tzerro",dimensionSet(0,0,0,1,0,0,0),0.0)
);


volScalarField C_zerro
(
    IOobject
    (
        "C_zerro",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,              // When using NO_READ we should initiliaze our variable, but I want it dependent og C_SiO and C_CO
        IOobject::NO_WRITE
    ),
    mesh,
    dimensionedScalar("C_zerro",dimensionSet(0,-3,0,0,1,0,0),0.0)
);


P_SiO_equ_1 = P_TOT / (1+ exp((-delta_G_1)/(R*Tg)));
P_SiO_equ_4 = P_TOT / (1+ exp((-delta_G_4)/(R*Tg)));
P_SiO_equ_2 = P_TOT*min(one, exp(delta_G_2/(2*R*Tg)));


R_1= k_1*r_C* max(C_C,C_zerro)*max((P_SiO - P_SiO_equ_1),zerro);

 R_2= k_2*exp(-E_2/(R*Ts))*max(P_SiO-P_SiO_equ_2,zerro);
R_neg2= k_neg2*exp(-E_neg2/(R*Ts))*max(C_SiO2_l,C_zerro)*max(C_Si, C_zerro);
R_3= k_3*max(C_SiO2_s,C_zerro)*max(Ts-Tm_SiO2,Tzerro);
 R_4= k_4*exp(-E_4/(R*Ts))*max(C_SiC,C_zerro)*max((P_SiO-P_SiO_equ_4),zerro);
 R_5= k_5*exp(-E_5/(R*Ts))*max(C_SiO2_l,C_zerro)*max(C_C,C_zerro);
```

A.1.8 themCond.H

```
//Script used to calculate thermal conduction for the solid/liquid and gas phase in order to calculate the heat exchange coefficient
/* A matrix has been made:

k (T) = b_i1T^4 + b_i2T + b_i3T + b_i4T +b_i5, [J/K kgmol] where i is a specie

C a11 a12 a13 a14 a15
SiC a21 a22 a23 a24 a25
SiO2_s a31 a32 a33 a34 a35
SiO2_l a41 a42 a43 a44 a45
Si a51 a52 a53 a54 a55
CO a61 a62 a63 a64 a65
SiO a71 a72 a73 a74 a75

This multiplied with:

T^4
T^3
T^2
T
1
*/

volScalarField kelvin
(
    IOobject
    (
        "kelvin",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::NO_WRITE
    ),
    mesh,
    dimensionedScalar("kelvin",dimensionSet(0,0,0,1,0,0,0),273) //[K]

);

volScalarField ks_C
(
    IOobject
    (
        "ks_C",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
```

```
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("ks_C",dimensionSet(1,1,-3,-1,0,0,0),0.0)
);


volScalarField ks_one
(
    IOobject
    (
        "ks_one",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::NO_WRITE
    ),
    mesh,
    dimensionedScalar("ks_one",dimensionSet(1,1,-3,-1,0,0,0),1.0)
);


volScalarField ks_SiC
(
    IOobject
    (
        "ks_SiC",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("ks_SiC",dimensionSet(1,1,-3,-1,0,0,0),0.0)
);
volScalarField ks_SiO2_s
(
    IOobject
    (
        "ks_SiO2_s",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("ks_SiO2_s",dimensionSet(1,1,-3,-1,0,0,0),0.0)
);


volScalarField ks_SiO2_l
(
    IOobject
    (
        "ks_SiO2_l",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("ks_SiO2_l",dimensionSet(1,1,-3,-1,0,0,0),0.0)
);


volScalarField ks_Si
(
    IOobject
    (
        "ks_Si",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("ks_Si",dimensionSet(1,1,-3,-1,0,0,0),0.0)
);

//Thermalconduction solid phase
ks_C = (b11 + b12*(Ts-kelvin))*(1-pow(eps_C,(eps_C_coeff))); //Must test this!

ks_SiC  = b21 + b22*(Ts-kelvin) + b23 * pow((Ts-kelvin), 2);
ks_SiO2_s = b31*pow(Ts,4) + b32*pow(Ts,3) + b33*pow(Ts,2) + b34*Ts + b35;//b31_const;//
ks_SiO2_l = b41*pow(Ts,4) + b42*pow(Ts,3) + b43*pow(Ts,2) + b44*Ts + b45;// b41_const;//
ks_Si = b51*pow(Ts,4) + b52*pow(Ts,3) + b53*pow(Ts,2) + b54*Ts + b55;
ks_T =  ks_C*(C_C/C_s)+ ks_SiC*(C_SiC/C_s)+ ks_SiO2_s*(C_SiO2_s/C_s)+ ks_SiO2_l*(C_SiO2_l/C_s)+ ks_Si*(C_Si/C_s);

//Thermal conduction gas-phase
```

```
volScalarField kg_CO = b61*pow(Tg,4) + b62*pow(Tg,3) + b63*pow(Tg,2) + b64*Tg + b65;
volScalarField kg_SiO = b71*pow(Tg,4) + b72*pow(Tg,3) + b73*pow(Tg,2) + b74*Tg + b75;
kg_T = kg_CO*(C_CO/C_g) + kg_SiO*(C_SiO/C_g);
```

A.1.9 viscosity.H

```
    //Create a scrift to calulate the dynamic viscocity of the the gas mixture
my=my_1*pow(Tg,4) + my_2*pow(Tg,3) + my_3*pow(Tg,2)+ my_4*pow(Tg,1) + my_5;
```

A.1.10 createFields.H

```
    volScalarField p
(IOobject(
        "p",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

volVectorField Ug

(
    IOobject
    (
        "Ug",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedVector("Ug",dimensionSet(0,1,-1,0,0,0,0),vector::zero)
);

volVectorField Us

(
    IOobject
    (
        "Us",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh

);

volVectorField Ug_m

(
    IOobject
    (
        "Ug_m",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh

);
volVectorField Us_m

(
    IOobject
    (
        "Us_m",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);


/*
Decleration of the velocity flux phi
```

```
it is a surface field (U is projected onto the face of each cell of the grid)
It is nessecarry when using divergence operator (fvm::div(phi,T))
*/

surfaceScalarField phig ("phig", linearInterpolate(Ug) & mesh.Sf());
surfaceScalarField phis ("phis", linearInterpolate(Us) & mesh.Sf());
surfaceScalarField phigm ("phigm", linearInterpolate(Ug_m) & mesh.Sf());
surfaceScalarField phism ("phism", linearInterpolate(Us_m) & mesh.Sf());



volScalarField Ts
(
    IOobject
    (
        "Ts",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
volScalarField Tg
(
    IOobject
    (
        "Tg",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

volScalarField C_C
(
    IOobject
    (
        "C_C",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

volScalarField C_SiC
(
    IOobject
    (
        "C_SiC",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);


volScalarField C_SiO2_s
(
    IOobject
    (
        "C_SiO2_s",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

volScalarField C_SiO2_l
(
    IOobject
    (
        "C_SiO2_l",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

```
volScalarField C_Si
(
    IOobject
    (
        "C_Si",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);


volScalarField C_CO
(
    IOobject
    (
        "C_CO",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);



volScalarField C_SiO
(
    IOobject
    (
        "C_SiO",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

volScalarField P_SiO
(
    IOobject
    (
        "P_SiO",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("P_SiO",dimensionSet(1,-1,-2,0,0,0,0),0.0)
);


volScalarField P_CO
(
    IOobject
    (
        "P_CO",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("P_CO",dimensionSet(1,-1,-2,0,0,0,0),0.0)
);


volScalarField h
(
    IOobject
    (
        "h",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("h",dimensionSet(1,-1,-3,-1,0,0,0),0.0)
);


volScalarField h_star
```

```
(
    IOobject
    (
        "h_star",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("h_star",dimensionSet(1,0,-3,-1,0,0,0),0.0)
);
volScalarField C_g
(
    IOobject
    (
        "C_g",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("C_g",dimensionSet(0,-3,0,0,1,0,0),0.0)
);


volScalarField C_s
(
    IOobject
    (
        "C_s",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("C_s",dimensionSet(0,-3,0,0,1,0,0),0.0)
);


volScalarField Cpspecific_s
(
    IOobject
    (
        "Cpspecific_s",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("Cpspecific_s",dimensionSet(0,2,-2,-1,0,0,0),0.0)
);

volScalarField Cpspecific_g
(
    IOobject
    (
        "Cpspecific_g",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("Cpspecific_g",dimensionSet(0,2,-2,-1,0,0,0),0.0)
);

volScalarField Q_el
(
    IOobject
    (
        "Q_el",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("Q_el",dimensionSet(1,-1,-3,0,0,0,0),0)
);

volScalarField R_1
(
    IOobject
    (
        "R_1",
```

```
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("R_1",dimensionSet(0,-3,-1,0,1,0,0),0.0)
);


volScalarField R_neg2
(
    IOobject
    (
        "R_neg2",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("R_neg2",dimensionSet(0,-3,-1,0,1,0,0),0.0)
);


volScalarField R_2
(
    IOobject
    (
        "R_2",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("R_2",dimensionSet(0,-3,-1,0,1,0,0),0.0)
);


volScalarField R_3
(
    IOobject
    (
        "R_3",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("R_3",dimensionSet(0,-3,-1,0,1,0,0),0.0)
);


volScalarField R_4
(
    IOobject
    (
        "R_4",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("R_4",dimensionSet(0,-3,-1,0,1,0,0),0.0)
);


volScalarField R_5
(
    IOobject
    (
        "R_5",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("R_5",dimensionSet(0,-3,-1,0,1,0,0),0.0)
);


volScalarField my
(
    IOobject
```

```
    (
        "my",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("my",dimensionSet(1,-1,-1,0,0,0,0),0.0)
);


volScalarField ks_T
(
    IOobject
    (
        "ks_T",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("ks_T",dimensionSet(1,1,-3,-1,0,0,0),0.0)
);

volScalarField kg_T
(
    IOobject
    (
        "kg_T",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("kg_T",dimensionSet(1,1,-3,-1,0,0,0),0.0)
);

volScalarField qs
(
    IOobject
    (
        "qs",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("qs",dimensionSet(1,-1,-3,0,0,0,0),0.0)
);


volScalarField qg
(
    IOobject
    (
        "qg",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("qg",dimensionSet(1,-1,-3,0,0,0,0),0.0)
);

volScalarField P_SiO_equ_1
(
    IOobject
    (
        "P_SiO_equ_1",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("P_SiO_equ_1",dimensionSet(1,-1,-2,0,0,0,0),0.0)
);


volScalarField P_SiO_equ_2
(
    IOobject
    (
        "P_SiO_equ_2",
```

```
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("P_SiO_equ_2",dimensionSet(1,-1,-2,0,0,0,0),0.0)
);


volScalarField P_SiO_equ_4
(
    IOobject
    (
        "P_SiO_equ_4",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("P_SiO_equ_4",dimensionSet(1,-1,-2,0,0,0,0),0.0)
);



//-----------------------------------
//-----------------------------------

Info<< "Reading transportProperties\n" << endl;

IOdictionary transportProperties
(
    IOobject
    (
        "transportProperties",
        runTime.constant(),
        mesh,
        IOobject::MUST_READ_IF_MODIFIED,
        IOobject::NO_WRITE
    )
);

//-----------------------------------



//Info<< "Reading permeability k\n" << endl;

dimensionedScalar k
(
    transportProperties.lookup("k")
);

//Info<< "Reading fluid viscosity mu\n" << endl;

dimensionedScalar mu
(
    transportProperties.lookup("mu")
);


//Info<< "Reading porosity eps\n" << endl;
dimensionedScalar eps
(
    transportProperties.lookup("eps")
);

//ks and kg are needed to calculate h ( heat exchange coefficient  in TEq)

//Info<< "Thermal conducticity of the solid, ks\n" << endl;
dimensionedScalar ks
(
    transportProperties.lookup("ks")
);

//Info<< "Thermal conducticity of the gas, kg\n" << endl;
dimensionedScalar kg
(
    transportProperties.lookup("kg")
);


//This is now calculated in a separate file!!!
//Info<< "Reading heat exchange coefficient h\n" << endl;
dimensionedScalar h_0
(
    transportProperties.lookup("h_0")
);
```

```
//Added 08.02.2019 for the Master Thesis


//Gas density  NB! We assume contant density for solid and gas phase!! :)

//Info<< "Density of the gas phase, rho_g\n" << endl;
dimensionedScalar rho_g
(
    transportProperties.lookup("rho_g")
);


//Info<< "Density of the solid-liquid phase, rho_s\n" << endl;
dimensionedScalar rho_s
(
    transportProperties.lookup("rho_s")
);


//Furnace and particle dimensions:


//Info<< "Heigh if the charge L, m\n" << endl;
dimensionedScalar L
(
    transportProperties.lookup("L")
);

//Info<< "Particle diameter dp\n" << endl;
dimensionedScalar dp
(
    transportProperties.lookup("dp")
);

//Info<< "Total pressure for reaction calculations, P_TOT\n" << endl;
dimensionedScalar P_TOT
(
    transportProperties.lookup("P_TOT")
);


//Info<< "Beta\n" << endl;
dimensionedScalar beta
(
    transportProperties.lookup("beta")
);


//Info<< "Prandtls numner, Pr\n" << endl;
dimensionedScalar Pr
(
    transportProperties.lookup("Pr")
);

//Values needed to calculate Q_el

//Info<< "Parameter calculatin Q_el, Q_el_a_val \n" << endl;
dimensionedScalar Q_el_a_val
(
    transportProperties.lookup("Q_el_a_val")
);

//Info<< "Parameter calculatin Q_el, Q_el_b_val \n" << endl;
dimensionedScalar Q_el_b_val
(
    transportProperties.lookup("Q_el_b_val")
);



//Q_el

/*
Info<< "Constants for calculations of heat generation from electrode, Q_el_a\n" << endl;
dimensionedScalar Q_el_a
(
    transportProperties.lookup("Q_el_a")
);
Info<< "Constants for calculations of heat generation from electrode, Q_el_b\n" << endl;
dimensionedScalar Q_el_b
(
    transportProperties.lookup("Q_el_b")
);
*/

//Parameters needed to calculate the mass balances
```

```
//---------------------------------------------------
//---------------------------------------------------
Info<< "Reading thermDat\n" << endl;

IOdictionary thermDat
(
    IOobject
    (
        "thermDat",
        runTime.constant(),
        mesh,
        IOobject::MUST_READ_IF_MODIFIED,
        IOobject::NO_WRITE
    )
);
//---------------------------------------------------


//Info<< "Gas constant R\n" << endl;
dimensionedScalar R
(
    thermDat.lookup("R")
);


//Info<< "Activation Energy for reaction 2, E_2\n" << endl;
dimensionedScalar E_2
(
    thermDat.lookup("E_2")
);

//Info<< "Activation Energy for reaction -2, E_neg2\n" << endl;
dimensionedScalar E_neg2
(
    thermDat.lookup("E_neg2")
);

//Info<< "Activation Energy for reaction 4, E_4\n" << endl;
dimensionedScalar E_4
(
    thermDat.lookup("E_4")
);

//Info<< "Activation Energy for reaction 5, E_5\n" << endl;
dimensionedScalar E_5
(
    thermDat.lookup("E_5")
);

//Values needed for Equilibrium Partial Pressure functions:

//Info<< "Parametervalue for Equilibrium Partial Pressure functions for reaction 1, a1\n" << endl;
dimensionedScalar a1
(
    thermDat.lookup("a1")
);

//Info<< "Parametervalue for Equilibrium Partial Pressure functions for reaction 1, b1\n" << endl;
dimensionedScalar b1
(
    thermDat.lookup("b1")
);

//Info<< "Parametervalue for Equilibrium Partial Pressure functions for reaction 1, c1\n" << endl;
dimensionedScalar c1
(
    thermDat.lookup("c1")
);


//Info<< "Parametervalue for Equilibrium Partial Pressure functions for reaction 2, a2\n" << endl;
dimensionedScalar a2
(
    thermDat.lookup("a2")
);


//Info<< "Parametervalue for Equilibrium Partial Pressure functions for reaction 2, b2\n" << endl;
dimensionedScalar b2
(
    thermDat.lookup("b2")
);

//Info<< "Parametervalue for Equilibrium Partial Pressure functions for reaction 2, c2\n" << endl;
dimensionedScalar c2
(
    thermDat.lookup("c2")
);
```

```
//Info<< "Parametervalue for Equilibrium Partial Pressure functions for reaction 4, a4\n" << endl;
dimensionedScalar a4
(
    thermDat.lookup("a4")
);


//Info<< "Parametervalue for Equilibrium Partial Pressure functions for reaction 4, b4\n" << endl;
dimensionedScalar b4
(
    thermDat.lookup("b4")
);

//Info<< "Parametervalue for Equilibrium Partial Pressure functions for reaction 4, c4\n" << endl;
dimensionedScalar c4
(
    thermDat.lookup("c4")
);

//Melting temperature of quartz

//Info<< "Melting temperature for quartz, Tm_SiO2\n" << endl;
dimensionedScalar Tm_SiO2
(
    thermDat.lookup("Tm_SiO2")
);



//--------------------------------------------------
//--------------------------------------------------
Info<< "Reading rateConstants\n" << endl;

IOdictionary rateConstants
(
    IOobject
    (
        "rateConstants",
        runTime.constant(),
        mesh,
        IOobject::MUST_READ_IF_MODIFIED,
        IOobject::NO_WRITE
    )
);
//--------------------------------------------------


//Info<< "Reactivity of carbon, r_C\n" << endl;
dimensionedScalar r_C
(
    rateConstants.lookup("r_C")
);

//Info<< "Reaction constant for reaction 1, k_1\n" << endl;
dimensionedScalar k_1
(
    rateConstants.lookup("k_1")
);


//Info<< "Reaction constant for reaction 2, k_2\n" << endl;
dimensionedScalar k_2
(
    rateConstants.lookup("k_2")
);

//Info<< "Reaction constant for reaction -2, k_neg2\n" << endl;
dimensionedScalar k_neg2
(
    rateConstants.lookup("k_neg2")
);

//Info<< "Reaction constant for reaction 3, k_3\n" << endl;
dimensionedScalar k_3
(
    rateConstants.lookup("k_3")
);


//Info<< "Reaction constant for reaction 4, k_4\n" << endl;
dimensionedScalar k_4
(
    rateConstants.lookup("k_4")
);


//Info<< "Reaction constant for reaction 5, k_5\n" << endl;
```

```
dimensionedScalar k_5
(
    rateConstants.lookup("k_5")
);



//---------------------------------------------------
//---------------------------------------------------
Info<< "Reading chemProperties\n" << endl;

IOdictionary chemProperties
(
    IOobject
    (
        "chemProperties",
        runTime.constant(),
        mesh,
        IOobject::MUST_READ_IF_MODIFIED,
        IOobject::NO_WRITE
    )
);
//---------------------------------------------------


//Chemical properties
/*
C 1
SiC 2
SiO2_s 3
SiO2_l 4
Si 5
CO 6
SiO 7
*/


//Info<< "Molar mass of C, Mm_C\n" << endl;
dimensionedScalar Mm_1
(
    chemProperties.lookup("Mm_1")
);

//Info<< "Molar mass of SiC, Mn_SiC\n" << endl;
dimensionedScalar Mm_2
(
    chemProperties.lookup("Mm_2")
);

//Info<< "Molar mass of SiO2(s), Mm_SiO2_l\n"  << endl;
dimensionedScalar Mm_3
(
    chemProperties.lookup("Mm_3")
);

//Info<< "Molar mass of SiO2(l), Mm_SiO2_l\n" << endl;
dimensionedScalar Mm_4
(
    chemProperties.lookup("Mm_4")
);

//Info<< "Molar mass of Si, Mm_Si\n" << endl;
dimensionedScalar Mm_5
(
    chemProperties.lookup("Mm_5")
);
//Info<< "Molar mass of CO, Mm_CO\n" << endl;
dimensionedScalar Mm_6
(
    chemProperties.lookup("Mm_6")
);
//Info<< "Molar mass of SiO, Mm_SiO\n" << endl;
dimensionedScalar Mm_7
(
    chemProperties.lookup("Mm_7")
);



//---------------------------------------------------------------
//---------------------------------------------------------------
Info<< "Reading enthalpyCalc\n" << endl;

IOdictionary enthalpyCalc
(
    IOobject
    (
```

```
        "enthalpyCalc",
        runTime.constant(),
        mesh,
        IOobject::MUST_READ_IF_MODIFIED,
        IOobject::NO_WRITE
    )
);
//---------------------------------------------------------


//Constants for enthalpy calculations are implemented in enthalpyCalc

/* Two matrixes have been made:


C a11 a12 a13 a14 a15
Sic a21 a22 a23 a24 a25
SiO2_s a31 a32 a33 a34 a35
SiO2_l a41 a42 a43 a44 a45
Si a51 a52 a53 a54 a55
CO a61 a62 a63 a64 a65
SiO a71 a72 a73 a74 a75

This multiplied with:

T^4
T^3
T^2
T
1

Yielding:

Cp_i (T) = a_i1T^4 + a_i2T + a_i3T + a_i4T +a_i5, where i is a specie
*/

////////////////////////////////////////////////////////////////////////
//Constant for heat capacity calculations:
////////////////////////////////////////////////////////////////////////


//Info<< "Constants for heat capacity calculations for C, a11 \n" << endl;
dimensionedScalar a11
(
    enthalpyCalc.lookup("a11")
);


//Info<< "Constants for heat capacity calculations for C, a12\n" << endl;
dimensionedScalar a12
(
    enthalpyCalc.lookup("a12")
);
//Info<< "Constants for heat capacity calculations for C, a13 \n" << endl;
dimensionedScalar a13
(
    enthalpyCalc.lookup("a13")
);
//Info<< "Constants for heat capacity calculations for C, a14 \n" << endl;
dimensionedScalar a14
(
    enthalpyCalc.lookup("a14")
);
//Info<< "Constants for heat capacity calculations for C, a15 \n" << endl;
dimensionedScalar a15
(
    enthalpyCalc.lookup("a15")
);

//SiC

//Info<< "Constants for heat capacity calculations for SiC, a21 \n" << endl;
dimensionedScalar a21
(
    enthalpyCalc.lookup("a21")
);


//Info<< "Constants for heat capacity calculations for SiC, a22\n" << endl;
dimensionedScalar a22
(
    enthalpyCalc.lookup("a22")
);
//Info<< "Constants for heat capacity calculations for SiC, a23 \n" << endl;
dimensionedScalar a23
(
    enthalpyCalc.lookup("a23")
);
//Info<< "Constants for heat capacity calculations for SiC, a24 \n" << endl;
```

```
dimensionedScalar a24
(
    enthalpyCalc.lookup("a24")
);
//Info<< "Constants for heat capacity calculations for SiC, a25 \n" << endl;
dimensionedScalar a25
(
    enthalpyCalc.lookup("a25")

);

//SiO2_s


//Info<< "Constants for heat capacity calculations for SiO2_s, a31 \n" << endl;
dimensionedScalar a31
(
    enthalpyCalc.lookup("a31")
);

//Info<< "Constants for heat capacity calculations for SiO2_s, a32 \n" << endl;
dimensionedScalar a32
(
    enthalpyCalc.lookup("a32")
);

//Info<< "Constants for heat capacity calculations for SiO2_s, a33 \n" << endl;
dimensionedScalar a33
(
    enthalpyCalc.lookup("a33")
);

//Info<< "Constants for heat capacity calculations for SiO2_s, a34 \n" << endl;
dimensionedScalar a34
(
    enthalpyCalc.lookup("a34")
);

//Info<< "Constants for heat capacity calculations for SiO2_s, a35 \n" << endl;
dimensionedScalar a35
(
    enthalpyCalc.lookup("a35")
);

//SiO2_l


//Info<< "Constants for heat capacity calculations for SiO2_l, a41 \n" << endl;
dimensionedScalar a41
(
    enthalpyCalc.lookup("a41")
);

//Info<< "Constants for heat capacity calculations for SiO2_l, a42 \n" << endl;
dimensionedScalar a42
(
    enthalpyCalc.lookup("a42")
);

//Info<< "Constants for heat capacity calculations for SiO2_l, a43 \n" << endl;
dimensionedScalar a43
(
    enthalpyCalc.lookup("a43")
);


//Info<< "Constants for heat capacity calculations for SiO2_l, a44 \n" << endl;
dimensionedScalar a44
(
    enthalpyCalc.lookup("a44")
);

//Info<< "Constants for heat capacity calculations for SiO2_l, a45\n" << endl;
dimensionedScalar a45
(
    enthalpyCalc.lookup("a45")
);

//Si

//Info<< "Constants for heat capacity calculations for Si, a51 \n" << endl;
dimensionedScalar a51
(
    enthalpyCalc.lookup("a51")
);

//Info<< "Constants for heat capacity calculations for Si, a52 \n" << endl;
dimensionedScalar a52
(
```

```
    enthalpyCalc.lookup("a52")
);

//Info<< "Constants for heat capacity calculations for Si, a53 \n" << endl;
dimensionedScalar a53
(
    enthalpyCalc.lookup("a53")
);
//Info<< "Constants for heat capacity calculations for Si, a54 \n" << endl;
dimensionedScalar a54
(
    enthalpyCalc.lookup("a54")
);
//Info<< "Constants for heat capacity calculations for Si, a55 \n" << endl;
dimensionedScalar a55
(
    enthalpyCalc.lookup("a55")
);


//CO


//Info<< "Constants for heat capacity calculations for CO, a61 \n" << endl;
dimensionedScalar a61
(
    enthalpyCalc.lookup("a61")
);

//Info<< "Constants for heat capacity calculations for CO, a62 \n" << endl;
dimensionedScalar a62
(
    enthalpyCalc.lookup("a62")
);
//Info<< "Constants for heat capacity calculations for CO, a63 \n" << endl;
dimensionedScalar a63
(
    enthalpyCalc.lookup("a63")
);
//Info<< "Constants for heat capacity calculations for CO, a64 \n" << endl;
dimensionedScalar a64
(
    enthalpyCalc.lookup("a64")
);
//Info<< "Constants for heat capacity calculations for CO, a65 \n" << endl;
dimensionedScalar a65
(
    enthalpyCalc.lookup("a65")
);


//SiO
//Info<< "Constants for heat capacity calculations for SiO, a71 \n" << endl;
dimensionedScalar a71
(
    enthalpyCalc.lookup("a71")
);

//Info<< "Constants for heat capacity calculations for SiO, a72 \n" << endl;
dimensionedScalar a72
(
    enthalpyCalc.lookup("a72")

);
//Info<< "Constants for heat capacity calculations for SiO, a73 \n" << endl;
dimensionedScalar a73
(
    enthalpyCalc.lookup("a73")
);

//Info<< "Constants for heat capacity calculations for SiO, a74 \n" << endl;
dimensionedScalar a74
(
    enthalpyCalc.lookup("a74")
);

//Info<< "Constants for heat capacity calculations for SiO, a75 \n" << endl;
dimensionedScalar a75
(
    enthalpyCalc.lookup("a75")
);


//////////////////////////////////////////////////////////////////////////
//Constant for heat enthalpy calculations:
//////////////////////////////////////////////////////////////////////////

/*
* A matrix has been made for the enthalpy calculations:
```

```
C c11 c12 c13 c14 c15
Sic c21 c22 c23 c24 c25
SiO2_s c31 c32 c33 c34 c35
SiO2_l c41 c42 c43 c44 c45
Si c51 c52 c53 c54 c55
CO c61 c62 c63 c64 c65
SiO c71 c72 c73 c74 c75

This multiplied with:

T^4
T^3
T^2
T
1

This yiels:
H_i (T) = c_i1T^4 + c_i2T + c_i3T + c_i4T +c_i5, where i is a specie

*/



//Info<< "Constants for heat capacity calculations for C, a11 \n" << endl;
dimensionedScalar c11
(
    enthalpyCalc.lookup("c11")
);

//Info<< "Constants for enthalpy calculations for C, a12\n" << endl;
dimensionedScalar c12
(
    enthalpyCalc.lookup("c12")
);
//Info<< "Constants for enthalpycalculations for C, a13 \n" << endl;
dimensionedScalar c13
(
    enthalpyCalc.lookup("c13")
);
//Info<< "Constants for enthalpy calculations for C, a14 \n" << endl;
dimensionedScalar c14
(
    enthalpyCalc.lookup("c14")
);
//Info<< "Constants for enthalpy calculations for C, a15 \n" << endl;
dimensionedScalar c15
(
    enthalpyCalc.lookup("c15")
);

//SiC

//Info<< "Constants for enthalpy calculations for SiC, a21 \n" << endl;
dimensionedScalar c21
(
    enthalpyCalc.lookup("c21")
);


//Info<< "Constants for enthalpycalculations for SiC, a22\n" << endl;
dimensionedScalar c22
(
    enthalpyCalc.lookup("c22")
);
//Info<< "Constants for enthalpy calculations for SiC, a23 \n" << endl;
dimensionedScalar c23
(
    enthalpyCalc.lookup("c23")
);
//Info<< "Constants for enthalpy calculations for SiC, a24 \n" << endl;
dimensionedScalar c24
(
    enthalpyCalc.lookup("c24")
);
//Info<< "Constants for enthalpy calculations for SiC, a25 \n" << endl;
dimensionedScalar c25
(
    enthalpyCalc.lookup("c25")

);

//SiO2_s


//Info<< "Constants for enthalpy calculations for SiO2_s, a31 \n" << endl;
dimensionedScalar c31
(
    enthalpyCalc.lookup("c31")
```

```
);

//Info<< "Constants for enthalpy calculations for SiO2_s, a32 \n" << endl;
dimensionedScalar c32
(
    enthalpyCalc.lookup("c32")
);

//Info<< "Constants for enthalpy calculations for SiO2_s, a33 \n" << endl;
dimensionedScalar c33
(
    enthalpyCalc.lookup("c33")
);

//Info<< "Constants for enthalpy calculations for SiO2_s, a34 \n" << endl;
dimensionedScalar c34
(
    enthalpyCalc.lookup("c34")
);

//Info<< "Constants for enthalpy calculations for SiO2_s, a35 \n" << endl;
dimensionedScalar c35
(
    enthalpyCalc.lookup("c35")
);

//SiO2_l


//Info<< "Constants for enthalpycalculations for SiO2_l, a41 \n" << endl;
dimensionedScalar c41
(
    enthalpyCalc.lookup("c41")
);

//Info<< "Constants for enthalpy calculations for SiO2_l, a42 \n" << endl;
dimensionedScalar c42
(
    enthalpyCalc.lookup("c42")
);

//Info<< "Constants for enthalpy calculations for SiO2_l, a43 \n" << endl;
dimensionedScalar c43
(
    enthalpyCalc.lookup("c43")
);


//Info<< "Constants for enthalpy calculations for SiO2_l, a44 \n" << endl;
dimensionedScalar c44
(
    enthalpyCalc.lookup("c44")
);

//Info<< "Constants for enthalpy calculations for SiO2_l, a45\n" << endl;
dimensionedScalar c45
(
    enthalpyCalc.lookup("c45")
);

//Si

//Info<< "Constants for enthalpy calculations for Si, a51 \n" << endl;
dimensionedScalar c51
(
    enthalpyCalc.lookup("c51")
);

//Info<< "Constants for enthalpy calculations for Si, a52 \n" << endl;
dimensionedScalar c52
(
    enthalpyCalc.lookup("c52")
);

//Info<< "Constants for enthalpy calculations for Si, a53 \n" << endl;
dimensionedScalar c53
(
    enthalpyCalc.lookup("c53")
);
//Info<< "Constants for enthalpy calculations for Si, a54 \n" << endl;
dimensionedScalar c54
(
    enthalpyCalc.lookup("c54")
);
//Info<< "Constants for enthalpy calculations for Si, a55 \n" << endl;
dimensionedScalar c55
(
    enthalpyCalc.lookup("c55")
);
```

```
//CO


//Info<< "Constants for enthalpy calculations for CO, a61 \n" << endl;
dimensionedScalar c61
(
    enthalpyCalc.lookup("c61")
);

//Info<< "Constants for enthalpy calculations for CO, a62 \n" << endl;
dimensionedScalar c62
(
    enthalpyCalc.lookup("c62")
);
//Info<< "Constants for enthalpy calculations for CO, a63 \n" << endl;
dimensionedScalar c63
(
    enthalpyCalc.lookup("c63")
);
//Info<< "Constants for enthalpy calculations for CO, a64 \n" << endl;
dimensionedScalar c64
(
    enthalpyCalc.lookup("c64")
);
//Info<< "Constants for enthalpy calculations for CO, a65 \n" << endl;
dimensionedScalar c65
(
    enthalpyCalc.lookup("c65")
);


//SiO
//Info<< "Constants for enthalpy calculations for SiO, a71 \n" << endl;
dimensionedScalar c71
(
    enthalpyCalc.lookup("c71")
);

//Info<< "Constants for enthalpy calculations for SiO, a72 \n" << endl;
dimensionedScalar c72
(
    enthalpyCalc.lookup("c72")

);
//Info<< "Constants for enthalpy calculations for SiO, a73 \n" << endl;
dimensionedScalar c73
(
    enthalpyCalc.lookup("c73")
);

//Info<< "Constants for enthalpy calculations for SiO, a74 \n" << endl;
dimensionedScalar c74
(
    enthalpyCalc.lookup("c74")
);

//Info<< "Constants for enthalpy calculations for SiO, a75 \n" << endl;
dimensionedScalar c75
(
    enthalpyCalc.lookup("c75")
);




//Reference enthalpy at T=298 K for all species: //NB! These must be found through HSC! Ask Merete Thangstad!


//Info<< "Reference temperature, T_ref \n" << endl;
dimensionedScalar T_ref
(
    enthalpyCalc.lookup("T_ref")
);


//Info<< "Standard enthalpy of reaction for C, Href_C \n" << endl;
dimensionedScalar Href_1
(
    enthalpyCalc.lookup("Href_1")
);

//Info<< "Standard enthalpy of reaction for SiC, Href_SiC\n" << endl;
dimensionedScalar Href_2
(
    enthalpyCalc.lookup("Href_2")
);
```

```
//Info<< "Standard enthalpy of reaction for SiO2_s, Href_SiO2_s \n" << endl;
dimensionedScalar Href_3
(
    enthalpyCalc.lookup("Href_3")
);


//Info<< "Standard enthalpy of reaction for SiO2_l, Href_SiO2_l \n" << endl;
dimensionedScalar Href_4
(
    enthalpyCalc.lookup("Href_4")
);


//Info<< "Standard enthalpy of reaction for Si, HrefS_Si \n" << endl;
dimensionedScalar Href_5
(
    enthalpyCalc.lookup("Href_5")
);

//Info<< "Standard enthalpy of reaction for CO, Href_CO \n" << endl;
dimensionedScalar Href_6
(
    enthalpyCalc.lookup("Href_6")
);


//Info<< "Standard enthalpy of reaction for SiO, Href_SiO \n" << endl;
dimensionedScalar Href_7
(
    enthalpyCalc.lookup("Href_7")
);
//Info<< "Standard enthalpy of reaction for SiO, H_SiO2_fus \n" << endl;
dimensionedScalar H_SiO2_fus
(
    enthalpyCalc.lookup("H_SiO2_fus")
);


//-----------------------------------------------------------
//-----------------------------------------------------------
Info<< "Reading dynamicViscosity\n" << endl;

IOdictionary dynamicViscosity
(
    IOobject
    (
        "dynamicViscosity",
        runTime.constant(),
        mesh,
        IOobject::MUST_READ_IF_MODIFIED,
        IOobject::NO_WRITE
    )
);
//-----------------------------------------------------------


//Info<< "Constants for dynamic viscisity calculations for CO(g) and SiO(g), my_1 \n" << endl;
dimensionedScalar my_1
(
    dynamicViscosity.lookup("my_1")
);


//Info<< "Constants for dynamic viscisity calculations for CO(g) and SiO(g), my_2 \n" << endl;
dimensionedScalar my_2
(
    dynamicViscosity.lookup("my_2")
);

//Info<< "Constants for dynamic viscisity calculations for CO(g) and SiO(g), my_3 \n" << endl;
dimensionedScalar my_3
(
   dynamicViscosity.lookup("my_3")
);

//Info<< "Constants for dynamic viscisity calculations for CO(g) and SiO(g), my_4 \n" << endl;
dimensionedScalar my_4
(
    dynamicViscosity.lookup("my_4")
);

//Info<< "Constants for dynamic viscisity calculations for CO(g) and SiO(g), my_5 \n" << endl;
dimensionedScalar my_5
(
    dynamicViscosity.lookup("my_5")
);
```

```
//---------------------------------------------------------------
//---------------------------------------------------------------
Info<< "Reading thermalCondCoeff\n" << endl;

IOdictionary thermalCondCoeff
(
    IOobject
    (
        "thermalCondCoeff",
        runTime.constant(),
        mesh,
        IOobject::MUST_READ_IF_MODIFIED,
        IOobject::NO_WRITE
    )
);
//---------------------------------------------------------------


//C
//Info<< "Constants for heat thermal conduction calculations for C, b11 \n" << endl;
dimensionedScalar b11
(
    thermalCondCoeff.lookup("b11")
);

//Info<< "Constants for heat thermal conduction calculations for C, b12 \n" << endl;
dimensionedScalar b12
(
    thermalCondCoeff.lookup("b12")
);


//Info<< "Porosity of carbon,eps_C\n" << endl;
dimensionedScalar eps_C
(
    thermalCondCoeff.lookup("eps_C")
);

dimensionedScalar eps_C_coeff
(
    thermalCondCoeff.lookup("eps_C_coeff")
);

//SiC


//Info<< "Constants for heat thermal conduction calculations for SiC, b21 \n" << endl;
dimensionedScalar b21
(
    thermalCondCoeff.lookup("b21")
);

//Info<< "Constants for heat thermal conduction calculations for SiC, b22 \n" << endl;
dimensionedScalar b22
(
    thermalCondCoeff.lookup("b22")
);

//Info<< "Constants for heat thermal conduction calculations for SiC, b23 \n" << endl;
dimensionedScalar b23
(
    thermalCondCoeff.lookup("b23")
);


//SiO2_s


dimensionedScalar b31_const
(
    thermalCondCoeff.lookup("b31_const")
);

//Info<< "Constants for heat thermal conduction calculations for SiO2_S, b31 \n" << endl;
dimensionedScalar b31
(
    thermalCondCoeff.lookup("b31")
);

//Info<< "Constants for heat thermal conduction calculations for SiO2_S, b32 \n" << endl;
dimensionedScalar b32
(
    thermalCondCoeff.lookup("b32")
);

//Info<< "Constants for heat thermal conduction calculations for SiO2_S, b33 \n" << endl;
dimensionedScalar b33
(
    thermalCondCoeff.lookup("b33")
```

```
);


//Info<< "Constants for heat thermal conduction calculations for SiO2_S, b34 \n" << endl;
dimensionedScalar b34
(
    thermalCondCoeff.lookup("b34")
);
//Info<< "Constants for heat thermal conduction calculations for SiO2_S, b35 \n" << endl;
dimensionedScalar b35
(
    thermalCondCoeff.lookup("b35")
);


//SiO2_l
dimensionedScalar b41_const
(
    thermalCondCoeff.lookup("b41_const")
);


//Info<< "Constants for heat thermal conduction calculations for SiO2_l, b41 \n" << endl;
dimensionedScalar b41
(
    thermalCondCoeff.lookup("b41")
);

//Info<< "Constants for heat thermal conduction calculations for SiO2_l, b42 \n" << endl;
dimensionedScalar b42
(
    thermalCondCoeff.lookup("b42")
);

//Info<< "Constants for heat thermal conduction calculations for SiO2_l, b43 \n" << endl;
dimensionedScalar b43
(
    thermalCondCoeff.lookup("b43")
);

//Info<< "Constants for heat thermal conduction calculations for SiO2_l, b44 \n" << endl;
dimensionedScalar b44
(
    thermalCondCoeff.lookup("b44")
);

//Info<< "Constants for heat thermal conduction calculations for SiO2_l, b45 \n" << endl;
dimensionedScalar b45
(
    thermalCondCoeff.lookup("b45")
);

//Si
//Info<< "Constants for heat thermal conduction calculations for Si, b51 \n" << endl;
dimensionedScalar b51
(
    thermalCondCoeff.lookup("b51")
);

//Info<< "Constants for heat thermal conduction calculations for Si, b52 \n" << endl;
dimensionedScalar b52
(
    thermalCondCoeff.lookup("b52")
);
//Info<< "Constants for heat thermal conduction calculations for Si, b53 \n" << endl;
dimensionedScalar b53
(
    thermalCondCoeff.lookup("b53")
);
//Info<< "Constants for heat thermal conduction calculations for Si, b54 \n" << endl;
dimensionedScalar b54
(
    thermalCondCoeff.lookup("b54")
);
//Info<< "Constants for heat thermal conduction calculations for Si, b55 \n" << endl;
dimensionedScalar b55
(
    thermalCondCoeff.lookup("b55")
);

//CO

//Info<< "Constants for thermal heat  conduction calculations for CO, b61 \n" << endl;
dimensionedScalar b61
(
    thermalCondCoeff.lookup("b61")
);

//Info<< "Constants for thermal heat  conduction calculations for CO, b62 \n" << endl;
```

```
dimensionedScalar b62
(
    thermalCondCoeff.lookup("b62")
);
//Info<< "Constants for thermal heat  conduction calculations for CO, b63 \n" << endl;
dimensionedScalar b63
(
    thermalCondCoeff.lookup("b63")
);

//Info<< "Constants for thermal heat  conduction calculations for CO, b64 \n" << endl;
dimensionedScalar b64
(
    thermalCondCoeff.lookup("b64")
);
//Info<< "Constants for thermal heat  conduction calculations for CO, b65 \n" << endl;
dimensionedScalar b65
(
    thermalCondCoeff.lookup("b65")
);

//SiO

//Info<< "Constants for thermal heat  conduction calculations for SiO, b71 \n" << endl;
dimensionedScalar b71
(
    thermalCondCoeff.lookup("b71")
);

//Info<< "Constants for thermal heat  conduction calculations for SiO, b72 \n" << endl;
dimensionedScalar b72
(
    thermalCondCoeff.lookup("b72")
);
//Info<< "Constants for thermal heat  conduction calculations for SiO, b73 \n" << endl;
dimensionedScalar b73
(
    thermalCondCoeff.lookup("b73")
);
//Info<< "Constants for thermal heat  conduction calculations for SiO, b74 \n" << endl;
dimensionedScalar b74
(
    thermalCondCoeff.lookup("b74")
);
//Info<< "Constants for thermal heat  conduction calculations for SiO, b75 \n" << endl;
dimensionedScalar b75
(
    thermalCondCoeff.lookup("b75")
);

#include "createFvOptions.H"
```

A.1.11 Make directory

file

```
    ergunMasterFoam.C

EXE = $(FOAM_USER_APPBIN)/ergunMasterFoam
```

options

```
    EXE_INC = \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/meshTools/lnInclude

EXE_LIBS = \
    -lfiniteVolume \
    -lfvOptions \
    -lmeshTools
```

A.2 Casefiles

A.2.1 Files in the 0 directory

C_C

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      C_C;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 -3 0 0 1 0 0]; //[kg m s K kgmol A cd]

internalField   uniform 7.075e4;//

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 7.075e4;
    }
    top
    {
        type zeroGradient;

    }

    bottom
    {
type zeroGradient;
    }


    frontAndBack
    {
        type            empty;
    }

}

// ************************************************************************* //
```

C_CO

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      C_C;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 -3 0 0 1 0 0]; //[kg m s K kgmol A cd]

internalField   uniform 7.075e4;//

boundaryField
{
```

```
    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 7.075e4;
    }
    top
    {
        type zeroGradient;

    }

    bottom
    {
type zeroGradient;
    }


    frontAndBack
    {
        type            empty;
    }

}

// ************************************************************************* //
```

C_CO

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      C_CO;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 -3 0 0 1 0 0]; //[kg m s K kgmol A cd]

internalField   uniform 5.849;

boundaryField
{
    inlet
    {
         type            fixedValue;
         value           uniform 5.849;
    }

    outlet
    {
        type            zeroGradient;
    }
    top
    {
        type zeroGradient;

    }

    bottom
    {
type zeroGradient;
    }


    frontAndBack
    {
        type            empty;
    }

}

// ************************************************************************* //
```

C_Si

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      C_Si;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 -3 0 0 1 0 0]; //[kg m s K kgmol A cd]

internalField   uniform 0; //0.1e4

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;//0.1e4
    }
    top
    {
        type zeroGradient;

    }

    bottom
    {
type zeroGradient;
    }


    frontAndBack
    {
        type            empty;
    }

}

// ************************************************************************* //
```

C_SiC

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      SiC;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 -3 0 0 1 0 0]; //[kg m s K kgmol A cd]

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
```

```
        type            fixedValue;
        value           uniform 0;
    }
    top
    {
        type zeroGradient;

    }

    bottom
    {
type zeroGradient;
    }

    frontAndBack
    {
        type            empty;
    }

}

// ************************************************************************* //
```

C_SiO

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      C_SiO;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 -3 0 0 1 0 0]; //[kg m s K kgmol A cd]

internalField   uniform 5.849;
boundaryField
{
    inlet
    {
         type            fixedValue;
         value           uniform 5.849;
    }

    outlet
    {
        type zeroGradient;
    }
    top
    {
        type zeroGradient;

    }

    bottom
    {
type zeroGradient;
    }


    frontAndBack
    {
        type            empty;
    }

}

// ************************************************************************* //
```

C_SiO2_l

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
```

```
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      C_SiO2_l;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 -3 0 0 1 0 0]; //[kg m s K kgmol A cd]

internalField   uniform 0;//0.1e4;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;//0.1e4;
    }
    top
    {
        type zeroGradient;

    }

    bottom
    {
type zeroGradient;
    }


    frontAndBack
    {
        type            empty;
    }

}

// ************************************************************************* //
```

C_SiO2_s

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      C_SiO2_s;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 -3 0 0 1 0 0]; //[kg m s K kgmol A cd]

internalField   uniform 2.152e4;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 2.152e4;
    }
    top
    {
        type zeroGradient;
```

```
    }

    bottom
    {
type zeroGradient;
    }


    frontAndBack
    {
        type            empty;
    }

}

// ************************************************************************* //
```

p

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0]; //[kg m s K kgmol A cd]

//internalField   uniform 101325;
internalField   nonuniform List<scalar>
100(
103325
103305
103285
103264
103244
103224
103204
103184
103163
103143
103123
103103
103083
103062
103042
103022
103002
102982
102961
102941
102921
102901
102881
102860
102840
102820
102800
102780
102759
102739
102719
102699
102679
102658
102638
102618
102598
102578
102557
102537
102517
102497
102477
102456
102436
```

```
102416
102396
102376
102355
102335
102315
102295
102274
102254
102234
102214
102194
102173
102153
102133
102113
102093
102072
102052
102032
102012
101992
101971
101951
101931
101911
101891
101870
101850
101830
101810
101790
101769
101749
101729
101709
101689
101668
101648
101628
101608
101588
101567
101547
101527
101507
101487
101466
101446
101426
101406
101386
101365
101345
101325

);
boundaryField
{
    inlet
    {
        type            fixedValue;
value uniform 103325;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 101325;
    }
    top
    {
        type zeroGradient;

    }

    bottom
    {
type zeroGradient;
    }


    frontAndBack
    {
        type            empty;
    }

}
```

```
// ************************************************************************* //
```

Tg

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      Tg;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   nonuniform List<scalar>
100(
2273
2253
2233
2213
2193
2173
2153
2133
2113
2093
2074
2054
2034
2014
1994
1974
1954
1934
1914
1894
1874
1854
1834
1814
1794
1774
1754
1734
1714
1694
1675
1655
1635
1615
1595
1575
1555
1535
1515
1495
1475
1455
1435
1415
1395
1375
1355
1335
1315
1295
1276
1256
1236
1216
1196
1176
1156
1136
1116
1096
1076
1056
```

```
1036
1016
996
976
956
936
916
896
877
857
837
817
797
777
757
737
717
697
677
657
637
617
597
577
557
537
517
497
478
458
438
418
398
378
358
338
318
298

);

boundaryField
{
    inlet
    {
        type            fixedValue;
value uniform   2273;
    }

    outlet
    {
        type            zeroGradient;
    }
    top
    {
        type zeroGradient;

    }

    bottom
    {
type zeroGradient;
    }


    frontAndBack
    {
        type            empty;
    }

}

// ************************************************************************* //
```

Ts

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
```

```
    format      ascii;
    class       volScalarField;
    object      Ts;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

//internalField   uniform 298;//


internalField   nonuniform List<scalar>
100
(2273
2253
2233
2213
2193
2173
2153
2133
2113
2093
2074
2054
2034
2014
1994
1974
1954
1934
1914
1894
1874
1854
1834
1814
1794
1774
1754
1734
1714
1694
1675
1655
1635
1615
1595
1575
1555
1535
1515
1495
1475
1455
1435
1415
1395
1375
1355
1335
1315
1295
1276
1256
1236
1216
1196
1176
1156
1136
1116
1096
1076
1056
1036
1016
996
976
956
936
916
896
877
857
837
817
797
777
```

```
757
737
717
697
677
657
637
617
597
577
557
537
517
497
478
458
438
418
398
378
358
338
318
298

)
;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
value uniform 298;
    }
     top
    {
        type zeroGradient;

    }
    bottom
    {
type zeroGradient;
    }

    frontAndBack
    {
        type            empty;
    }

}

// ************************************************************************* //
```

Ug_m

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      Ug_m;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (2.35297 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
```

```
value uniform (2.35297 0 0);
    }

    outlet
    {
        type             zeroGradient;
    }
     top
    {
        type noSlip;

    }

    bottom
    {
type noSlip;
    }

    frontAndBack
    {
        type            empty;
    }

}

// ************************************************************************* //
```

Us_m

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      Us_m;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (-0.000103376 0 0);

boundaryField
{
    inlet
    {
        type             zeroGradient;
    }

    outlet
    {
        type            fixedValue;
value uniform (-0.000103376 0 0);
    }
     top
    {
        type noSlip;

    }

    bottom
    {
type noSlip;
    }

    frontAndBack
    {
        type            empty;
    }

}

// ************************************************************************* //
```

Us

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      Us;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (-6.25e-5 0 0);

boundaryField
{
    inlet
    {
        type             zeroGradient;
    }

    outlet
    {
        type            fixedValue;
value uniform (-6.25e-5 0 0);
    }
     top
    {
        type noSlip;

    }

    bottom
    {
type noSlip;
    }

    frontAndBack
    {
        type            empty;
    }

}

// ************************************************************************* //
```

A.2.2 Files in constant folder

chemProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      chemProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

//Chemical properties:
/*
C 1
SiC 2
SiO2_s 3
SiO2_l 4
Si 5
CO 6
SiO 7
*/
```

```
//[kg m s K mol A cd] Tatt fra SI-Chemical Data
Mm_1 Mm_1 [1 0 0 0 -1 0 0] 12.01e-3; //[kg/mol]
Mm_2 Mm_2 [1 0 0 0 -1 0 0] 40.1e-3;
Mm_3 Mm_3 [1 0 0 0 -1 0 0] 60.09e-3 ;
Mm_4 Mm_4 [1 0 0 0 -1 0 0] 60.09e-3;
Mm_5 Mm_5 [1 0 0 0 -1 0 0] 28.09e-3;
Mm_6 Mm_6 [1 0 0 0 -1 0 0] 28.01e-3;
Mm_7 Mm_7 [1 0 0 0 -1 0 0] 44.09e-3;


// ************************************************************************* //
```

dynamicViscosity

```
   /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      dynamicViscosity;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
//[kg m s K kgmol A cd]
//my(T)=my_1*T + my_2T + my_3T + my_4*T^1 + my_5
//This is kinematic viscosity
my_1           my_1 [1 -1 -1 -4 0 0 0] 4.507e-17;
my_2           my_2 [1 -1 -1 -3 0 0 0] -1.378e-13 ;
my_3           my_3 [1 -1 -1 -2 0 0 0] 2.067e-10;
my_4           my_4 [1 -1 -1 -1 0 0 0] 7.980e-10;
my_5           my_5 [1 -1 -1 0 0 0 0]  -3.632e-08;

//my(T)=my_1*T + my_2T + my_3T + my_4*T^1 + my_5
// ************************************************************************* //
```

enthalpyCalc

```
   /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      enthalpyCalc;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
/* A matrix has been made:

Cp_i (T) = a_i1T^4 + a_i2T + a_i3T + a_i4T +a_i5, [J/K kgmol] where i is a specie

C a11 a12 a13 a14 a15
SiC a21 a22 a23 a24 a25
SiO2_s a31 a32 a33 a34 a35
SiO2_l a41 a42 a43 a44 a45
Si a51 a52 a53 a54 a55
CO a61 a62 a63 a64 a65
SiO a71 a72 a73 a74 a75

This multiplied with:

T^4
T^3
T^2
T
1
*/
```

```
//[kg m s K mol A cd]
//C
a11 a11 [1 2 -2 -5 -1 0 0]-1.948e-12 ; //[kg m/s K mol]
a12 a12 [1 2 -2 -4 -1 0 0] 1.501e-08; //[kg m/s K mol]
a13 a13 [1 2 -2 -3 -1 0 0] -4.292e-05; //[kg m/s K mol]
a14 a14 [1 2 -2 -2 -1 0 0] 0.05617; //[kg m/s K mol]
a15 a15 [1 2 -2 -1 -1 0 0] -4.595; //[kg m/s K mol]

//SiC
a21 a21 [1 2 -2 -5 -1 0 0] -2.858e-12;
a22 a22 [1 2 -2 -4 -1 0 0] 2.244e-08;
a23 a23 [1 2 -2 -3 -1 0 0] -6.475e-05;
a24 a24 [1 2 -2 -2 -1 0 0] 0.08443;
a25 a25 [1 2 -2 -1 -1 0 0] 8.93;

//SiO2_s
a31 a31 [1 2 -2 -5 -1 0 0] -1.37e-11;
a32 a32 [1 2 -2 -4 -1 0 0] 8.93e-08;
a33 a33 [1 2 -2 -3 -1 0 0] -2.03e-04;
a34 a34 [1 2 -2 -2 -1 0 0] 0.1957;
a35 a35 [1 2 -2 -1 -1 0 0] 2.567;


//SiO1_l
a41 a41 [1 2 -2 -5 -1 0 0] 0;
a42 a42 [1 2 -2 -4 -1 0 0] 0;
a43 a43 [1 2 -2 -3 -1 0 0] 0;
a44 a44 [1 2 -2 -2 -1 0 0] 0;
a45 a45 [1 2 -2 -1 -1 0 0] 83.5;

//Si
a51 a51 [1 2 -2 -5 -1 0 0] 0;
a52 a52 [1 2 -2 -4 -1 0 0] 0;
a53 a53 [1 2 -2 -3 -1 0 0] 0;
a54 a54 [1 2 -2 -2 -1 0 0] 0;
a55 a55 [1 2 -2 -1 -1 0 0] 27.20;

//CO
a61 a61 [1 2 -2 -5 -1 0 0] 4.204e-13;
a62 a62 [1 2 -2 -4 -1 0 0] -2.541e-09;
a63 a63 [1 2 -2 -3 -1 0 0] 3.557e-06;
a64 a64 [1 2 -2 -2 -1 0 0] 0.004230;
a65 a65 [1 2 -2 -1 -1 0 0] 27.33;

//SiO
a71 a71 [1 2 -2 -5 -1 0 0] -5.632e-12;
a72 a72 [1 2 -2 -4 -1 0 0] 4.008e-08;
a73 a73 [1 2 -2 -3 -1 0 0] -0.0001025;
a74 a74 [1 2 -2 -2 -1 0 0] 0.1161;
a75 a75 [1 2 -2 -1 -1 0 0] 7.694;


////////////////////////////////////////////////////////
//Enthalpy calculations
////////////////////////////////////////////////////////


//C
c11 c11 [1 2 -2 -4 -1 0 0] 6.841e-10;
c12 c12 [1 2 -2 -3 -1 0 0] -5.314e-06;
c13 c13 [1 2 -2 -2 -1 0 0] 0.01607;
c14 c14 [1 2 -2 -1 -1 0 0] 2.523;
c15 c15 [1 2 -2 0 -1 0 0]  -2223;


//SiC
c21 c21 [1 2 -2 -4 -1 0 0] 1.038e-09;
c22 c22 [1 2 -2 -3 -1 0 0] -7.975e-06;
c23 c23 [1 2 -2 -2 -1 0 0] 0.02374;
c24 c24 [1 2 -2 -1 -1 0 0] 20.03;
c25 c25 [1 2 -2 0 -1 0 0]  -80032;

//SiO2_s
c31 c31 [1 2 -2 -4 -1 0 0] -3.707e-09;
c32 c32 [1 2 -2 -3 -1 0 0] 2.130e-05;
c33 c33 [1 2 -2 -2 -1 0 0] -0.03319;
c34 c34 [1 2 -2 -1 -1 0 0] 86.73;
c35 c35 [1 2 -2 0 -1 0 0]  -936747;

//SiO1_l
c41 c41 [1 2 -2 -4 -1 0 0] -3.707e-09;
c42 c42 [1 2 -2 -3 -1 0 0] 2.130e-05;
c43 c43 [1 2 -2 -2 -1 0 0] -0.03319;
c44 c44 [1 2 -2 -1 -1 0 0] 86.73;
c45 c45 [1 2 -2 0 -1 0 0]  -936747;

//Si
c51 c51 [1 2 -2 -4 -1 0 0] 0;
c52 c52 [1 2 -2 -3 -1 0 0] 0;
```

```
c53 c53 [1 2 -2 -2 -1 0 0] 0;
c54 c54 [1 2 -2 -1 -1 0 0] 27.2;
c55 c55 [1 2 -2 0 -1 0 0]  40605;

//CO
c61 c61 [1 2 -2 -4 -1 0 0] 5.128e-11;
c62 c62 [1 2 -2 -3 -1 0 0] -9.048e-07;
c63 c63 [1 2 -2 -2 -1 0 0] 0.005021;
c64 c64 [1 2 -2 -1 -1 0 0] 25.53;
c65 c65 [1 2 -2 0 -1 0 0]  -118526;

//SiO
c71 c71 [1 2 -2 -4 -1 0 0] 9.445e-10;
c72 c72 [1 2 -2 -3 -1 0 0] -6.937e-06;
c73 c73 [1 2 -2 -2 -1 0 0] 0.02078;
c74 c74 [1 2 -2 -1 -1 0 0] 30.27;
c75 c75 [1 2 -2 0 -1 0 0]  -111621;

T_ref T_ref [0 0 0 1 0 0 0] 298.73; //[K]

//Enthalpy
Href_1 Href_1 [1 2 -2 0 -1 0 0] 0; //[J/kmol]=[kgm/skmol]
Href_2 Href_2 [1 2 -2 0 -1 0 0] -71.902e6;
Href_3 Href_3 [1 2 -2 0 -1 0 0] -910.857e6;
Href_4 Href_4 [1 2 -2 0 -1 0 0] 9.565e6;
Href_5 Href_5 [1 2 -2 0 -1 0 0] 48.715e6;
Href_6 Href_6 [1 2 -2 0 -1 0 0] -110.541e6;
Href_7 Href_7 [1 2 -2 0 -1 0 0] -100.4e6;

H_SiO2_fus H_SiO2_fus [1 2 -2 0 -1 0 0] 11;
// ************************************************************************* //
```

rateConstants

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      rateConstants;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
//[kg m s K kgmol A cd]

/*
Reactions:

[1] SiO(g) + 2C(s) -> SiC(s) + CO(g) R_1
[2] 2SiO(g) = SiO2(l) + Si(l) R_2 and R_neg2
[3] SiO2(s) -> SiO2(l) R_3
[4] SiO(g) + SiC(s) -> 2Si(l) + CO(g) R_4
[5] SiO2(l) + C(s) -> SiO(g) + CO(g) R_5

Reaction rates: R_i = k_i*exp(Ei/RT)....;
- Both reaction rates and activation energies need to be defined

The units of each reaction constant has been carefully chosen so that the overall reaction rates have units [mol/m^3 s]

R_1=k1*r_C*C_C*(P_sio-P_sio_equ);
R_2=k2*exp()*(Psio-Psio,equ);
R_neg2= k_neg2*exp()*C_SiO2(l)*C_Si
R_3=k3*C_SiO2(s)*(T-Tm)
R_4=k4*exp()*C_SiC*(PSiO-PSiO,eq)
R_5=k_5*exp()*C_SiO(l)*C_C
*/


k_1 k_1 [-1 1 1 0 0 0 0] 4.8e-10; //[sm/kg]
k_2 k_2 [-1 -2 1 0 1 0 0] 9.8e-6; //[mol s/m kg]
k_neg2       k_neg2 [0 3 -1 0 -1 0 0] 0; //[m/mol s]
k_3 k_3 [0 0 -1 -1 0 0 0] 5.9e-7; //[1/sK]
k_4 k_4 [-1 1 1 0 0 0 0]  1.1e-7; //[sm/kg]
k_5 k_5 [0 3 -1 0 -1 0 0] 7.07e-11; //[m/mol s]
r_C r_C [0 0 0 0 0 0 0 ] 0.1; //[-]
// ************************************************************************* //
```

thermalCondCoeff

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      thermalCondCoeff;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


//C
b11            b11 [1 1 -3 -1 0 0 0] 0.973;
b12            b12 [1 1 -3 -2 0 0 0] 6.34e-3;
eps_C        eps_C [0 0 0 0 0 0 0] 0.5;
eps_C_coeff eps_C_coeff [0 0 0 0 0 0 0] 2/3;

//SiC
b21            b21 [1 1 -3 -1 0 0 0] 1.2 ;
b22            b22 [1 1 -3 -2 0 0 0] 0.002;
b23            b23 [1 1 -3 -3 0 0 0] 0.62e-4;

//SiO2_s
b31_const b31_const [1 1 -3 -1 0 0 0] 1.167;

//[kg m s K kgmol A cd]
b31            b31 [1 1 -3 -5 0 0 0] 8.791e-12;//-7.10e-12;
b32            b32 [1 1 -3 -4 0 0 0] -4.26336e-08;//1.27e-08;
b33            b33 [1 1 -3 -3 0 0 0] 7.63158e-05;//7.53e-06;
b34            b34 [1 1 -3 -2 0 0 0] -0.05935617;//-2.37e-02;
b35            b35 [1 1 -3 -1 0 0 0] 19.38849298;//12.95;


b41_const b41_const [1 1 -3 -1 0 0 0] 1.167;

b41            b41 [1 1 -3 -5 0 0 0] 8.791e-12;//-7.10e-12;
b42            b42 [1 1 -3 -4 0 0 0] -4.26336e-08;//1.27e-08;
b43            b43 [1 1 -3 -3 0 0 0] 7.63158e-05;//7.53e-06;
b44            b44 [1 1 -3 -2 0 0 0] -0.05935617;//-2.37e-02;
b45            b45 [1 1 -3 -1 0 0 0] 19.38849298;//12.95;


//Si
b51            b51 [1 1 -3 -5 0 0 0] 3.16e-09; //Project thesis
b52            b52 [1 1 -3 -4 0 0 0] -1.28e-05; //Project thesis
b53            b53 [1 1 -3 -3 0 0 0] 0.0196; //Project thesis
b54            b54 [1 1 -3 -2 0 0 0] -13.676; //Project thesis
b55            b55 [1 1 -3 -1 0 0 0] 4069.3; //Project thesis

//CO
b61            b61 [1 1 -3 -5 0 0 0] -1.00e-17;
b62            b62 [1 1 -3 -4 0 0 0] 9.80e-11;
b63            b63 [1 1 -3 -3 0 0 0] -3.57e-08;
b64            b64 [1 1 -3 -2 0 0 0] 9.13e-05;
b65            b65 [1 1 -3 -1 0 0 0] 2.94e-04;


//SiO
b71            b71 [1 1 -3 -5 0 0 0] -1.00e-17;
b72            b72 [1 1 -3 -4 0 0 0] 9.80e-11;
b73            b73 [1 1 -3 -3 0 0 0] -3.57e-08;
b74            b74 [1 1 -3 -2 0 0 0] 9.13e-05;
b75            b75 [1 1 -3 -1 0 0 0] 2.94e-04;

// ************************************************************************* //
```

thermDat

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
```

```
    format      ascii;
    class       dictionary;
    location    "constant";
    object      thermDat;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
//[kg m s K mol A cd]
R             R [1 2 -2 -1 -1 0 0] 8.314;
Tm_SiO2 Tm_SiO2 [0 0 0 1 0 0 0] 1996.0;

//Activation energy
E_2           E_2 [1 2 -2 0 -1 0 0] 0;
E_neg2     E_neg2 [1 2 -2 0 -1 0 0] 0;
E_4           E_4 [1 2 -2 0 -1 0 0] 0;
E_5           E_5 [1 2 -2 0 -1 0 0] 0;

//Parametervalue for Equilibrium Partial Pressure functions for reaction i
//[kg m s K kgmol A cd]

a1 a1 [1 2 -2 0 -1 0 0] -8.183e4;
a2 a2 [1 2 -2 0 -1 0 0] -8.178e5;
a4 a4 [1 2 -2 0 -1 0 0] -9.901e4;

b1 b1 [1 2 -2 -1 -1 0 0] -1.687;
b2 b2 [1 2 -2 -1 -1 0 0] -9.903e1;
b4 b4 [1 2 -2 -1 -1 0 0] -1.264e2;


c1 c1 [1 2 -2 -1 -1 0 0] 1.528e1;
c2 c2 [1 2 -2 -1 -1 0 0] 1.141e3;
c4 c4 [1 2 -2 -1 -1 0 0] 1.019e3;

// ************************************************************************* //
```

transportProperties

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
mu            mu [1 -1 -1 0 0 0 0] 4.847e-5;
k              k [0 2 0 0 0 0 0] 7.815e-8;//9.41e-8;
h_0 h_0 [1 -1 -3 -1 0 0 0] 200;
eps          eps [0 0 0 0 0 0 0] 0.3;
rho_s      rho_s [1 -3 0 0 0 0 0] 2.481e3;
rho_g      rho_g [1 -3 0 0 0 0 0] 0.23;
dp            dp [0 1 0 0 0 0 0] 0.08;
P_TOT      P_TOT [1 -1 -2 0 0 0 0] 101325;
L              L     [0 1 0 0 0 0 0] 2.5;
ks       ks [1 1 -3 -1 0 0 0] 3.65;
kg             kg [1 1 -3 -1 0 0 0] 0.096;
beta     beta [0 0 0 0 0 0 0 ] 10;
Pr     Pr [0 0 0 0 0 0 0 ] 0.7;
Q_el_a_val    Q_el_a_val [1 -1 -3 0 0 0 0] 10;
Q_el_b_val    Q_el_b_val [1 -1 -3 0 0 0 0] 5;

// ************************************************************************* //
```

A.2.3 System directory

blockMeshDich

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
```

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;
Lx 2.5;
Ly 0.1;
Lz 0.1;

vertices
(
    (0   0   0)
    ($Lx 0   0)
    ($Lx $Ly 0)
    (0   $Ly 0)
    (0   0   $Lz)
    ($Lx 0   $Lz)
    ($Lx $Ly $Lz)
    (0   $Ly $Lz)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (100 1 1) simpleGrading (1 1 1)
);

edges
(
);

boundary //Top and bottom were added 12.03!! Might be a mistake here!
(
    inlet
    {
        type patch;
        faces
        (
            (0 4 7 3)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (2 6 5 1)
        );
    }
    top
{
type wall;
faces
(
(3 7 6 2)
);
}

bottom
{
type wall;
faces
(
(1 5 4 0)
);
}


    frontAndBack
    {
        type empty;
        faces
        (
            //(1 5 4 0)
    //(3 7 6 2)
    (0 3 2 1)
    (4 5 6 7)
        );
    }
);


/*
mergePatchPairs
(
```

```
);
*/
// ************************************************************************* //
```

controlDict

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     ergunMasterFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         80400;//1.5 runthrough

deltaT         1;//0001;//0.00001;//0.000001;

writeControl    runTime;

writeInterval   1200;//write out every 20 min

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;


// ************************************************************************* //
```

fvSchemes

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}


gradSchemes
{
```

```
    default         Gauss linear;
    grad(p)         Gauss linear;
}


//Might want a multivaiant selection, check OpenFoam.org userguide!! It is good to use multivariateSelection so that all the species have
//exists for grouping multiple equation terms together, and applying the same limiters on all terms, using the strongest limiter calculated for all ter
// A good example of this is in a set of mass transport equations for fluid species, where it is good practice to apply the same discretisation to all
divSchemes
{
    default         none;
    div(phig,Tg) Gauss linearUpwind grad(Tg);
    div(phis,Ts) Gauss linearUpwind grad(Ts);
    div(phism,C_C) Gauss linearUpwind grad(C_C);
    div(phism,C_SiC) Gauss linearUpwind grad(C_SiC);
    div(phism,C_SiO2_s) Gauss linearUpwind grad(C_SiO2_s);
    div(phism,C_SiO2_l) Gauss linearUpwind grad(C_SiO2_l);
    div(phism,C_Si) Gauss linearUpwind grad(C_Si);
    div(phigm,C_CO) Gauss linearUpwind grad(C_CO);
    div(phigm,C_SiO) Gauss linearUpwind grad(C_SiO);
}

laplacianSchemes
{
    default         none;
    laplacian((k|mu),p) Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
p ;
}

// ************************************************************************* //
```

fvSolution

```
    /*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0;
    }

     Us
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0;
    }

    Ts
    {
        solver          PBiCG;
        preconditioner  DILU;
```

```
        tolerance       1e-06;
        relTol          0;
    }

    Tg
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-06;
        relTol          0;
    }

    C_C
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-06;
        relTol          0;
    }

    C_SiC
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-06;
        relTol          0;
    }

    C_SiO2_s
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-06;
        relTol          0;
    }

    C_SiO2_l
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-06;
        relTol          0;
    }

    C_Si
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-06;
        relTol          0;
    }

    C_CO
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-06;
        relTol          0;
    }

    C_SiO
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-06;
        relTol          0;
    }
}
SIMPLE
{
    nNonOrthogonalCorrectors 2;
}
// ************************************************************************* //
```

Appendix B

High Temperature Model

As the two models are very similar, only the changes needed to do from the LT model in order to obtain the HT model is presented here.

B.1 Solver

The only needed changes int the solver are adding the file elHeatSource.H and changing the temperature equation in ergunMasterFoam.C as shown in the following sections and including the elHeatSoruce.H file. And not calculating the gas velcity in the solver.

B.1.1 ergunMasterFoam.C

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Copyright (C) 2011-2018 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.


\*---------------------------------------------------------------------------*/

#include "fvCFD.H"
#include "fvOptions.H"  //30.10
#include "simpleControl.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{
    #include "setRootCaseLists.H"
    #include "createTime.H"
    #include "createMesh.H"
    simpleControl simple(mesh);
    #include "createFields.H"
    #include "pressureCalc.H"
```

```
    #include "viscosity.H"
    #include "themCond.H"

    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

   // Info<< "\nCalculating preassure distribution\n" << endl;

    while (simple.loop(runTime))
    {
        Info<< "Time = " << runTime.timeName() << nl << endl;

 while (simple.correctNonOrthogonal())
        {
           solve
   (
fvm::laplacian(k/mu, p)
   );
        }


#include "viscosity.H" //Need to calculate viscosity before velocity for ergun!!!!


//Ug=-k/mu*fvc::grad(p);
Ug.correctBoundaryConditions();


phig = linearInterpolate(Ug) & mesh.Sf(); //The surface flux phi is updated from the new value of the velocity profile Ug
phis = linearInterpolate(Us) & mesh.Sf(); //The surface flux phi is updated from the new value of the velocity profile Us


phigm = linearInterpolate(Ug_m) & mesh.Sf(); //The surface flux phi is updated from the new value of the velocity profile Ug_m
phism = linearInterpolate(Us_m) & mesh.Sf(); //The surface flux phi is updated from the new value of the velocity profile Us_m



#include "reactionRates.H"
#include "reactionHeatGas.H"
#include "elHeatSource.H"
#include "themCond.H"
//#include "heatExchangeCoeffCalc.H"

fvScalarMatrix TgEqn
(
eps*rho_g*Cpspecific_g*fvm::ddt(Tg) +rho_g*Cpspecific_g*fvm::div(phig,Tg)
==
eps*qg-fvm::Sp(h_0,Tg)+h_0*Ts+0.1*Q_el
);
TgEqn.solve();

fvScalarMatrix TsEqn
(
(1.-eps)*rho_s*Cpspecific_s*fvm::ddt(Ts) + rho_s*Cpspecific_s*fvm::div(phis,Ts)
==
(1-eps)*qs-fvm::Sp(h_0,Ts)+h_0*Tg+0.9*Q_el
);
TsEqn.solve();
#include "massBalance.H"
#include "pressureCalc.H"

runTime.write();

        Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
            << "  ClockTime = " << runTime.elapsedClockTime() << " s"
            << nl << endl;
    }

    Info<< "End\n" << endl;

    return 0;
}

// ************************************************************************* //
```

B.1.2 elHeatSource.H

```
//Info<< "Solver: elHeatSource.H \n" << endl;

volScalarField Q_el_a
(
    IOobject
    (
        "Q_el_a",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
```

```
        IOobject::NO_WRITE
    ),
    mesh,
    dimensionedScalar("Q_el_a",dimensionSet(1,-1,-3,0,0,0,0),200) //[Q_el]= W/m=kg/sm;
);


volScalarField Q_el_b
(
    IOobject
    (
        "Q_el_b",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::NO_WRITE
    ),
    mesh,
    dimensionedScalar("Q_el_b",dimensionSet(1,-1,-3,0,0,0,0),5) //[Q_el]= W/m=kg/sm;

);

volScalarField oneEl
(
    IOobject
    (
        "oneEl",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,              // When using NO_READ we should initiliaze our variable, but I want it dependent og C_SiO and C_CO
        IOobject::NO_WRITE
    ),
    mesh,
    dimensionedScalar("oneEl",dimensionSet(1,-1,-3,0,0,0,0),1.0)
);

volVectorField centres = Us.mesh().C();
volScalarField x = centres.component(0);
          Info<< min(x) << nl << endl;
          Info<< max(x) << nl << endl;

Q_el=Q_el_a*(-1+exp((Q_el_b/oneEl)*(1-x/L)));
```

B.2 Case files

As the case files are the same and only some parameters were changed the file code is not presented here. The main changes are that the gas velocity was initialized in the zero folder, the height of the model was adjusted to 1 m and concentrations of C(s) and SiC(s) was changed according to the high temperature base case. The code can be provided upon request.

Fatbardha Hajdini

Dynamic Modelling of the Silicon Arc Furnace

NTNU
Norwegian University of
Science and Technology