Numerical Simulation and Optimization of the Alumina Distribution in an Aluminium Electrolysis Pot

THÈSE N^{O} 5023 (2011)

PRÉSENTÉE LE 6 MAI 2011
À LA FACULTÉ SCIENCES DE BASE
CHAIRE D'ANALYSE ET DE SIMULATION NUMÉRIQUE
PROGRAMME DOCTORAL EN MATHÉMATIQUES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Thomas HOFER

acceptée sur proposition du jury:

Prof. B. Dacorogna, président du jury
Prof. J. Rappaz, directeur de thèse
Dr Y. Caratini, rapporteur
Prof. M. Rappaz, rapporteur
Prof. R. Touzani, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2011

Abstract

This thesis is about the numerical simulation and optimization of the alumina repartition in the bath of an aluminium electrolysis pot.

A mathematical model is set up which contains the feeding of alumina particles to the bath, the dissolution of the particles, the motion of the alumina in the bath, and the consumption of the alumina by electrolysis. The arising convection-diffusion equations are solved using stabilized finite elements. Different weak formulations of the convection-diffusion equation are compared and we find a formulation which ensures mass conservation even if the velocity field is not divergence free. A comparison of the complete numerical simulation with real world experiments is carried out, showing that the numerical results are in very good agreement with the measured values.

In a second part we optimize the feeding of the particles, with the aim of getting a more uniform distribution of the alumina concentration in the bath. A mathematical model of the optimization problem is proposed and subsequently solved using particle swarm optimization and Newton's method.

Keywords: Finite element method, convection-diffusion equation, mass conservation, aluminium electrolysis, particle dissolution, numerical simulation, numerical optimization.

Résumé

Le sujet de cette thèse est la simulation et l'optimisation numériques de la répartition de l'alumine dans le bain d'une cuve d'électrolyse d'aluminium.

Un modèle mathématique est introduit qui contient l'alimentation du bain avec des particules d'alumine, la dissolution des particules, le mouvement de l'alumine dans le bain et la consommation de l'alumine par l'électrolyse. Les équations de convection-diffusion qui apparaissent dans le modèle sont résolues en utilisant une approximation par des éléments finis stabilisés. Différentes formulations faibles de l'équation de convection-diffusion sont comparées afin de trouver une formulation qui assure la conservation de la masse même en présence d'un champ de vitesse qui n'est pas à divergence nulle. Une comparaison de la simulation numérique complète avec des expériences réelles est accomplie, montrant que les résultats numériques sont en très bon accord avec les valeurs mesurées.

Dans la deuxième partie du travail nous considérons l'optimisation de l'alimentation du bain, dans le but d'obtenir une distribution de l'alumine plus uniforme dans le bain. Nous proposons un modèle mathématique du problème d'optimisation qui est résolu en utilisant l'optimisation par essaims particulaires et la méthode de Newton.

Mots clés: Méthode des éléments finis, équation de convection-diffusion, conservation de la masse, électrolyse d'aluminium, dissolution de particules, simulation numérique, optimisation numérique.

Acknowledgements

I am grateful to Professor Jacques Rappaz who offered me the opportunity to do this PhD thesis, who supported me during the work and gave me helpful advices, but who also gave me the freedom to try things on my own. I thank Jacques for all the things I learned from him, which was a lot more than just mathematics.

I also thank Yves Caratini, Michel Rappaz and Rachid Touzani for having accepted to be members of the jury of my thesis, for having read my thesis and for the helpful remarks they have made. I thank Bernard Dacorogna for being president of the jury.

I would also like to thank the Rio Tinto Alcan Company for the financial support, and particularly the team at the Laboratoire de Recherche de Fabrication in Saint-Jean-de-Maurienne for their interest in my work and their helpful inputs.

I am deeply indebted to Michel Flück for endless support with programming and the computer in general, and with alucell in particular, and also for his patience and his interest in other topics than my work. I am grateful to Marco Picasso for his support in theoretical questions, and for the good collaboration we had regarding his teaching.

I thank all my colleagues that I had at the ASN for the good times we had together, namely Aleš, Pierre, Vicky, Christophe, Chantal, Guillaume, Wissam, Laurent, Nicolas, Sébastien, Alexandre C. and M., Romain, Viljami, Imbo. In particular, I thank Stéphane for the good ambiance we had in the office, and Gilles for coaching me in different things and for supporting me during long meetings.

I also thank all my friends elsewhere who always reminded me of the fact that there are other things in life than work.

Finally, I am very grateful to my family who supported me in many ways all along this work.

Contents

Introduction **1**

I Simulation of the dissolution of alumina **7**

1 Model **9**
1.1 Physical model . . . 9
1.2 Mathematical model . . . 10
1.3 Computation of the velocity field . . . 16

2 A convection-diffusion equation **21**
2.1 Model problem and desired properties of the solution . . . 21
2.2 Direct weak formulation . . . 22
2.3 Simple mass conserving formulation . . . 23
2.4 Simple L^2-stable formulation . . . 24
2.5 Velocity projection on sine and cosine functions . . . 24
2.6 Projection on divergence free finite element basis . . . 26
2.7 Helmholtz decomposition of the velocity field . . . 27
2.8 Meanfree mass conserving formulation . . . 28
2.9 Meanfree L^2-stable mass conserving formulation . . . 30
2.10 Conclusion . . . 33

3 Discretization **35**
3.1 Discretization in time . . . 35
3.2 Discretization in R . . . 36
3.3 Discretization in space . . . 37

4 Numerical aspects **41**
4.1 Particle dissolution . . . 41
4.2 Liquid alumina consumption . . . 44

5 Numerical Results **47**
5.1 Convergence test with known solution . . . 47
5.2 Comparison of different formulations . . . 48
5.3 Simulation of a real case . . . 51

6 Validation of the numerical solution **61**
6.1 Description . . . 61
6.2 Comparison with the first experiment . . . 62
6.3 Comparison with the second experiment . . . 68

6.4 Conclusion 68

7 Conclusion **71**

II Optimization **73**

8 Problem formulation **75**
8.1 Physical problem 75
8.2 Short introduction to optimization 76
8.3 Mathematical formulation of the optimization problem 76

9 Optimization algorithms **81**
9.1 Newton's method and line search 81
9.2 Automatic differentiation 83
9.3 Particle swarm optimization 87
9.4 Comparison of Newton's method and PSO 91

10 Numerical results **93**
10.1 Comparison of the norm for the objective function 93
10.2 Optimization of the feeder positions 96
10.3 Optimization of the load weights 101
10.4 Discussion of the final time of the simulation 102

11 Conclusion **105**

Bibliography **107**

Introduction

Aluminium is the most abundant metal in the Earth's crust (8.3% by weight) [18], but because of its strong affinity to oxygen it is rare in its free form. In the Earth's crust bauxite is the most important ore of aluminium. With the Bayer process bauxite is refined to alumina, an aluminium oxyde (Al_2O_3). Aluminium is then extracted by electrolysis, in the so called Hall-Héroult process. In this process, alumina is dissolved in an electrolytic bath and then reduced to aluminium. The overall reaction of this electrolysis is

$$2\mathrm{Al_2O_3} + 3\mathrm{C} \rightarrow 4\mathrm{Al} + 3\mathrm{CO_2}.$$

A schematic representation of the electrolysis pot is shown on Figures 1 and 2.

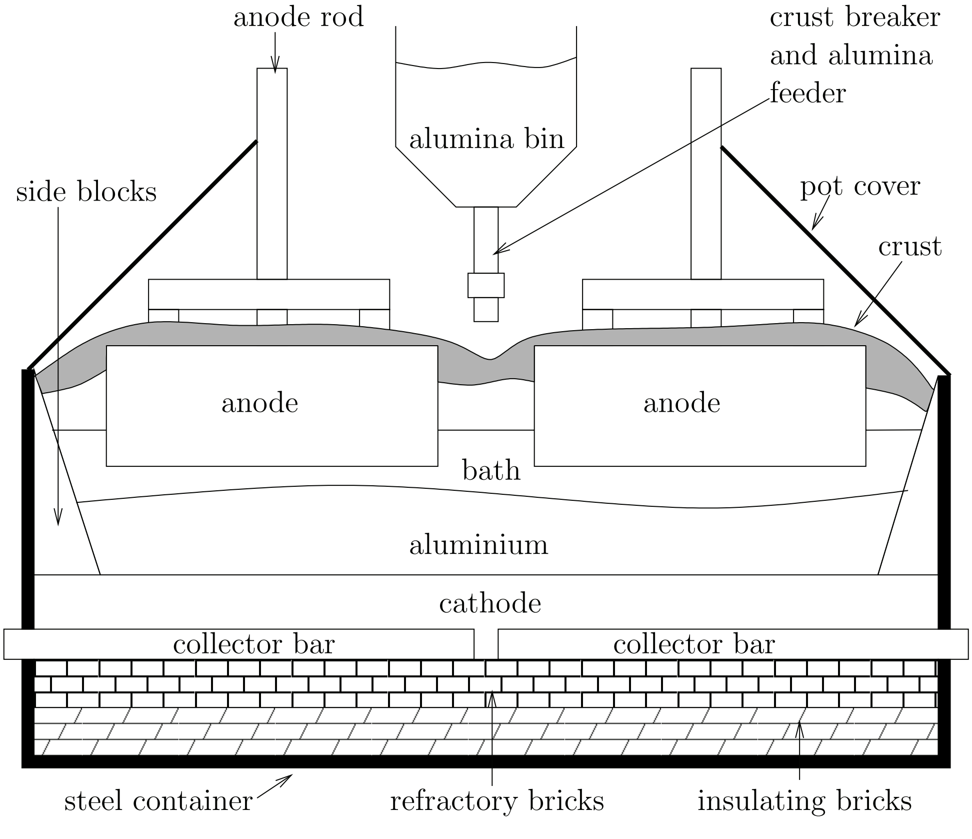


Figure 1: Schematic representation of an aluminium reduction pot. Inspired by [2].

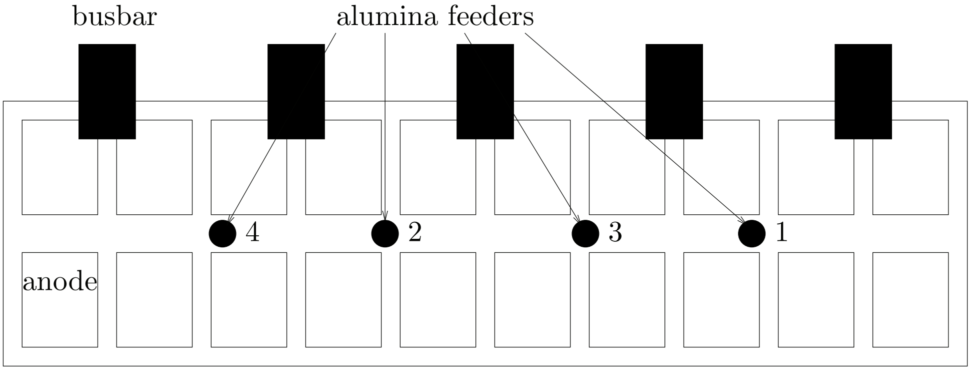


Figure 2: Schematic representation of the aluminium cell seen from above, with the alumina feeders and the feeding order during one feeding cycle.

The Hall-Héroult process

We will now present a description of the Hall-Héroult process. For more details we refer to the work by Grjotheim et al. [20].

The alumina reduction occurs in a vessel, which consists of several parts. There is an outer steel shell, containing the whole setup. On the bottom, we have some layers of thermally insulating bricks to reduce heat losses from the bath. On top of these, we have some layers of refractory bricks which are very resistant to the high cell temperatures. The molten bath and aluminium are in a container made of carbon. The bottom part of the carbon container is called cathode, even if from an electrochemical point of view it is the molten aluminium which acts as the cathode. Under each cathode there is an iron bar, called collector bar, which transports the current out of the cell.

The bath and the aluminium are liquid due to the high temperature, and they are separated because of their different densities. The chemical reactions for the alumina reduction occur in the molten bath. The electrical current necessary for the electrolysis is transported to the bath through the carbon anodes which are partially immersed into the bath. Anode rods, which are mostly made of aluminium and hold the anodes, are fixed to a large structure. The bridge leads the current to the anodes and makes it possible to raise or lower all of the anodes at the same time. In this way the cell voltage can be controlled. Since the carbon of the anodes takes part in the chemical reaction, the anodes are slowly consumed and have to be replaced regularly. The anodes cover almost the whole surface of the bath. On top of them and covering also the channels between the anodes and between the anodes and the boundary of the cell is a protective layer, consisting of solidified bath and alumina, to prevent heat losses. It is called crust.

The CO_2 created during the reaction escapes from the bath as gas bubbles. Since most of the surface of the bath is covered by the anodes, a bubble layer of CO_2 is built underneath the anodes, in the bath. The aluminium formed by the reduction of alumina sinks to the bottom and increases the height of the liquids in the cell as the production goes on. The aluminium at the bottom has to be pumped out regularly. During the electrolysis, the alumina dissolved in the bath is consumed and has to be restored periodically. Therefore, the cell is equipped with an alumina bin and a feeding system which delivers alumina to the electrolyte. The feeding system consists of a crust breaker and a feeder. Every time alumina has to be added to the bath, the crust breaker, which is basically a steel rod, opens a hole in the crust. Then, the feeder

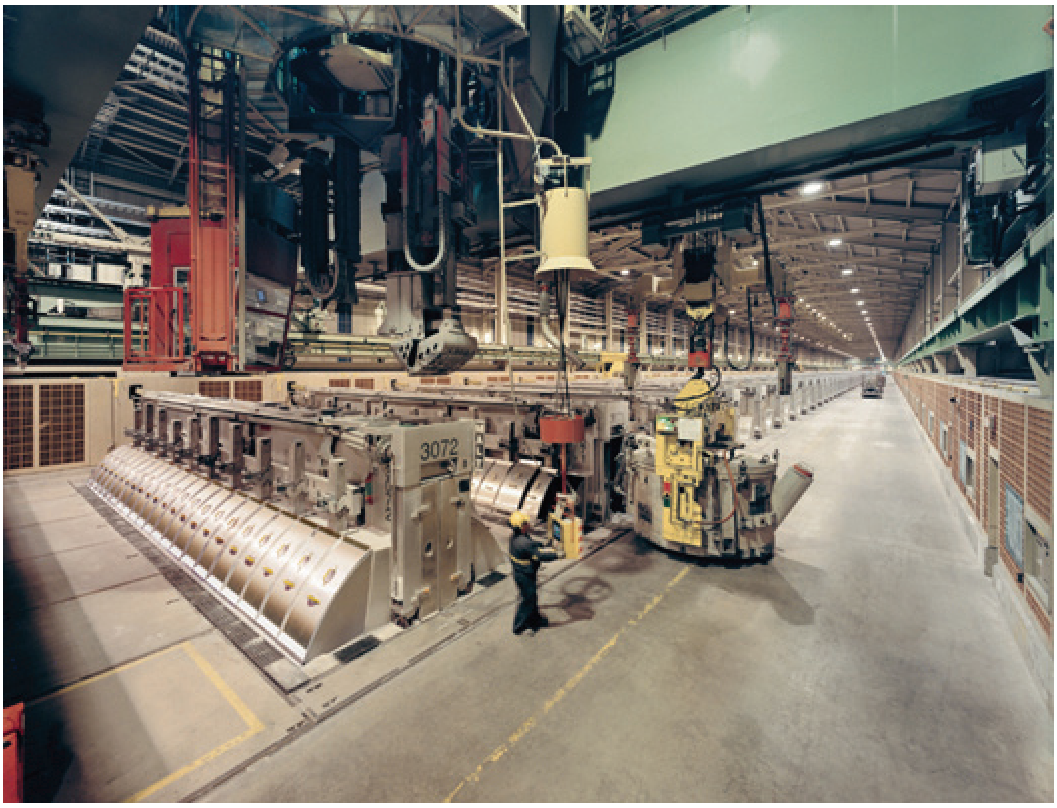


Figure 3: View of a potroom. Siphoning of aluminium is done at the right. Photo Courtesy of Rio Tinto Alcan.

dumps a predefined amount of alumina into the bath. Depending on the size of the pot, there are several crust breakers and feeders, usually 4 to 5. Fumes, which escape from the pot crust, are collected and directed to a gas treatment centre. To prevent the fumes from escaping, the cells are completely closed, partly by removable pot covers. The cells are about 15 metres long, 3 metres large, and 1 metre high. In an aluminium production plant, there are usually hundreds of aluminium cells connected electrically in series. The current is supplied to the cells through heavy aluminium busbars. A picture of a potroom is shown in Figure 3. There we also see how the aluminium produced is siphoned off.

In order to keep the bath at liquid state, the temperature of the bath has to be at about 960°C. This temperature is created by the electric resistance (Joule effect), mainly in the bath, but also in the anodes, the cathodes and the metal pad. To reach the right temperature with a current of about 350 kA, the height of the bath under the anode is only about 4 cm. The electric resistance is partly due to the gas bubble layers underneath the anodes, and depends also on the alumina concentration in the bath.

The liquid aluminium and the electrolytic bath are moving because they are subject to electromagnetic forces. These electromagnetic forces (Lorentz forces) arise from the combination of the current density running through the fluids, and the magnetic induction due to the electric configuration of a hall containing many cells.

The efficiency of an aluminium reduction cell depends mainly on the alumina concentration in the bath. This concentration should be between 1.5 and 3 weight percents (wt%). The current efficiency measures the percentage of current which is used for the aluminium reduction. It is usually about 95%, the rest of the current being used mostly for the inverse reaction. It is not possible to know the current efficieny exactly, and therefore it is not possible to know how much alumina is consumed. Hence, it is difficult to add the right amount of alumina to the bath, while this is crucial. If the alumina concentration becomes too high, the added solid alumina does not dissolve anymore and sinks to the bottom of the cell, forming sludge and disturbing the current flow with decreased current efficiency as a result. On the other hand, if the alumina concentration becomes too low the cell goes into a state called anode effect. In this state, the normal aluminium producing reactions are interrupted and other reactions take place, creating gases such as CF_4 (PFC-14, tetrafluoromethane) and C_2F_6 (PFC-116, hexafluoroethane). These gases adhere to the bottom of the anodes and form an insulating layer, which then provokes an increase of the cell voltage. Thus, we have a decrease of the current efficiency, and at the same time creation of green house gases which have a more than 1000 times higher global warming potential than CO_2 [15, section 2.10].

Avoiding the anode effect is therefore very important both economically and ecologically. To this end it is not sufficient to control the overall alumina concentration in the bath, but also the local concentration, which should be between 1.5 and 3 [wt%]. For this reason there are several alumina feeders in each cell, and the moving liquids help additionally distributing the alumina concentration.

The aluminium production process is energy-intensive. With todays technology, the overall energy efficiency of the electrolysis is about 50%, with excess energy being dissipated as heat loss from the cell. To produce one ton of aluminium about 13.5 MWh are needed. Thus, for a production site producing 100'000 tons of aluminium per year, a power of about 154 MW is needed, which is roughly one sixth of the power of a standard nuclear power plant. World production of primary aluminium in 2008 was 39.4 million tons [9, p.8].

Optimization of the aluminium production

Under such conditions, aluminium companies try to optimize their pots. For example, they want to increase the amperage of the cell, since the aluminium produced is proportional to the current, at least with constant current efficiency. Higher amperage does not mean that less energy is consumed per unit of aluminium. But there is an economic gain through reduced capital expenditure per unit annual production [38]. More current in a cell implies that the distance between the anodes and the metal pad has to be decreased, otherwise the heating by Joule effect is too important. This decrease of the anode - metal pad distance comes with a number of issues, one being less efficient dissolving and mixing of the alumina in the bath. It is therefore important to choose good configurations for the alumina feeders.

As pointed out before, the alumina feeding is done at several locations. As an example, on Figure 2 we have four feeder positions. A feeding cycle length has to be defined, which is usually about one minute. During each feeding cycle, every feeder dumps a predefined amount of alumina into the bath. The order of the feeding could be chosen as on Figure 2, and the feeding times are normally uniformly distributed over the cycle. Hence, if the cycle length is one minute, feeding takes place at 0, 15, 30 and 45 seconds. Then the feeding cycle is repeated. The feeder configuration consists thus in the feeder positions, the feeding time of each feeder during one feeding cycle, the cycle length, and the amount of alumina added by each feeder. The total amount of alumina added has to match the quantity of alumina transformed into aluminium.

Finding optimal feeder configurations requires knowing the alumina repartition in the bath for given feeder configurations. But it is not easy to accurately measure the alumina concentration in the bath. The liquids are at 960°C, they are covered by the anodes and the crust, the bath is very corrosive, and there are strong magnetic fields near the pots, all of which limits considerably the choice of measurement equipment. Then, to have a good idea of the concentration distribution in the bath, it is necessary to make measurements over the whole bath at the same time, without disturbing the electrolytic process too much since this should go on during the whole measurement. At last, since the aim is to change things in order to see whether the concentration gets more uniform, one has to make modifications to the production process, which could also be rather complicated if for example the feeder positions should be varied. All these factors make real experiments complicated and time consuming, and thus expensive, and the results are probably not very precise.

Aim of this work

Hence, other methods to improve aluminium production could be more useful. One other method is numerical simulation. Here, one tries to make a physical and then a mathematical model of the most important processes. The mathematical problem arising is then solved by numerical methods. The advantage is that there are fewer persons involved and that the technology used is essentially limited to computing power, which makes the numerical simulation rather cheap. The problem of this solution is certainly that it is strongly dependent on the quality of the model, i.e., on the simplifications and assumptions made. Also, the arising mathematical problem could be too complicated to be solved reasonably well within a reasonable time, although both the computers and the numerical methods become more and more powerful. Finally, numerical simulation will not be able to completely replace real world experiments. Simulation results can only serve as guidelines and must be taken with some care. But these guidelines can be very useful and greatly reduce the number of experiments to reach a certain result.

The aim of this work is to find optimal feeder configurations which give the most uniform concentration in the bath, using numerical methods. We will proceed in two steps. In a first step, we will set up a simulation of the addition of the alumina to the bath, the dissolution of the solid alumina, the repartition of the liquid alumina concentration and the decrease of concentration due to electrolysis. Then, based on this simulation, we will optimize the feeder configuration to reach the most uniform alumina concentration in the bath possible.

The simulation of the concentration depends very much on the steady-state velocity field in the bath. The velocity field which will be used is obtained by numerical methods. Since it influences the position of the interface between the bath and the metal pad, the electrical potential of the cell, the current density and thus the induced magnetic field, all these quantities have also to be computed when computing the velocity field. Therefore, the complete model to compute the velocities of the liquids is quite involved, see [14] for a detailed description.

Organisation of the document

The present document is separated in two parts. The first part is devoted to the simulation of the alumina concentration in the bath. We will start in Chapter 1 by describing the physical and mathematical model that will be used. Solid and liquid alumina will be treated separately. Simplifications and details not contained in the description of the overall aluminium production process above will be given. Since the velocity field is important in our model, we describe at the end of the chapter the algorithm to compute the interface position and the velocity field. In the second chapter, we will consider a model convection-diffusion problem. We state some properties that the numerical solution of this problem should show. Then, we introduce several different numerical schemes to compute the solution to the convection-diffusion problem, and for every scheme we verify if the desired properties are satisfied. We discretize the equations from the mathematical model in Chapter 3, using the numerical scheme that we judged most appropriate in the second chapter. We continue in Chapter 4 by showing numerical details of the different parts of the simulation. Numerical results will be presented in the fifth chapter. There, we will start by showing that the method converges in the correct theoretical order. We compare the solutions obtained using the various numerical schemes presented in the second chapter, and we do a long term computation to see whether our solution converges to some realistic periodic solution. In Chapter 6, we compare our results with real world experiments. We finish the first part by conclusions.

In the second part, we will optimize the alumina feeder configuration, using the simulation of the first part. We will, in Chapter 8, describe the precise problem and put it in a mathematical form. In Chapter 9, we present the general optimization algorithms we are going to use. We show numerical results in Chapter 10. There, we start by comparing different formulations of the objective functions and we justify our choice. We continue by showing results of the optimization of the feeder positions and the load weights. At the end of this chapter, we look at the influence of the final time of the simulation on our optimization results. The second part ends with conclusions concerning the optimization.

Part I

Simulation of the dissolution of alumina

Chapter 1

Model

The first chapter is devoted to the modeling of the dissolution of alumina particles in the electrolytic bath. Having in mind the description of the Hall-Héroult process given in the introduction, we will present in the first section the physical model of our simulation problem. We then give the mathematical equations which approximate this model in Section 1.2. Some important properties of the model are shown at the end of the second section. Since our model relies heavily on the velocity field present in the bath, we give in the third section a description of how the velocity field is numerically computed. This helps to understand some numerical properties of the velocity and the interface between the bath and the metal pad.

1.1 Physical model

In the aluminium production pot, alumina is found in the electrolytic bath. When the alumina is added to the bath, it is in a solid state, in the form of a powder. We will model this powder as a set of spherical particles with different radii. Initially the particle mass-size distribution must fulfil some requirements set up by the aluminium production company. In particular, the function ensures that the amount of very small particles is limited, because these particles tend to be blown away during feeding because of the heat. The maximal size of the particles is about 200 μm.

The particles are added at several locations between the anodes in the central channel. Compared to the width and length of the bath, the central channel is rather narrow, and therefore it contains only few vertices of the mesh. For this reason, we add the particles under the anodes in the bath. They are then transported by the moving bath. The bath moves due to the strong electric current which passes through the bath, and which induces electromagnetical forces on the liquid aluminium and on the bath. The computation of the velocity field is not part of this thesis. We will use discrete velocity fields computed in a steady-state algorithm which was developed by J. Descloux, M. Flueck and M. V. Romerio, and which is described in [34, Section 4.1]. The part of the algorithm which computes the velocity will be described in Section 1.3. The velocity field is incompressible and there is no outflow or inflow at the boundaries of the bath. We suppose that the particles are transported along streamlines of the velocity field at the same speed and without affecting them, since the particles are very small. For the same reason, the buoyancy force on the particles is small and we do not take it into account.

The particles dissolve while being transported by the bath. The dissolution kinetics are not completely known, i.e., it is not known whether the particles dissolve because of a chemical reaction, diffusion, or some other reason. Specialists think that the main reason for the dissolution is the chemical reaction. We will also show a mathematical model which is based on diffusion as the only dissolution kinetics.

The alumina particles dissolve and what appears will then be modeled as a liquid alumina concentration. When the particles dissolve, they are chemically transformed into ions, but we will always speak about liquid alumina concentration, or only concentration, for simplicity. This concentration is again convected by the moving bath and it is slightly diffusing at the same time. The diffusion coefficient of alumina in the bath is about $1.5 \cdot 10^{-9}$ m^2/sec, while the velocity of the bath is about 0.1 m/sec. However, the velocity field is supposed to be turbulent, at a scale which is not resolved by the mesh. Therefore, engineers propose to use a turbulent diffusion coefficient, which is in the range $[9.4 \cdot 10^{-7}, 4.7 \cdot 10^{-4}]$ $[\mathrm{m}^2/\mathrm{s}]$(see [21]). In the present work we always use a diffusion coefficient of $D = 5 \cdot 10^{-4}$ $[\mathrm{m}^2/\mathrm{s}]$, unless noted otherwise. We will suppose that the alumina concentration has no influence on the rest of the cell. This implies that we do not recompute the current or the velocity field based on the concentrations in the bath, and we suppose that the current efficiency remains constant.

The last part that we model is the electrolysis, which consumes the liquid alumina concentration. Electrolysis takes place at the interfaces bath-anode and bath - cathode. In our case the interface bath-cathode is replaced by the interface bath - metal pad, since liquid aluminium is a good electrical conductor and acts therefore like the cathode for the bath. We know that alumina, respectively the ions that were created during the dissolution, is not directly transformed into aluminium, but the precise chemical reactions which transform alumina into aluminium are not known. We therefore assume that the electrolytic process takes place in the whole bath, i.e., the alumina concentration diminishes in the whole bath due to the electrolysis. We will work with two different models, one assuming uniform consumption over the whole bath, the other depending on the current density which passes through the bath. The aluminium that is created by the electrolysis sinks to the bottom of the pot because it has a higher density than the bath. In our model the newly formed aluminium is neglected.

1.2 Mathematical model

We will denote the domain of the bath by $\Omega_{\mathrm{el}} \subset \mathbb{R}^3$. Let $c = c(\vec{x}, t)$ be the liquid alumina concentration depending on $\vec{x} \in \Omega_{\mathrm{el}}$ and the time t, and $n_p = n_p(\vec{x}, R, t)$ the alumina particle density, which additionally depends on R, the radius of the particles. The physical unit of the concentration is $[\mathrm{mol} \cdot \mathrm{m}^{-3}]$ and for the particle density it is $[(\text{number of particles}) \cdot \mathrm{m}^{-4}]$.

The model can be essentially described by two equations, one for the particle density and one for the concentration. The particle density follows a convection equation [6],

$$\frac{\partial n_p(\vec{x}, R, t)}{\partial t} + \vec{u}(\vec{x}).\vec{\nabla}_x n_p(\vec{x}, R, t) + \frac{\partial}{\partial R}(n_p(\vec{x}, R, t) f(R, c(\vec{x}, t)) = S(\vec{x}, R, t). \tag{1.2.1}$$

Here, $\vec{u}$ is the steady-state velocity of the bath, and S stands for a source due to the addition of the particles by the feeders. The function $f(R, c)$ describes the dissolution rate of alumina particles with radius R in a liquid with concentration c. Formally we have

$$\dot{R}(t) = f(R(t), c(\vec{x}, t)) \tag{1.2.2}$$

which gives the dissolution dynamics at position $\vec{x}$. The source S and the dissolution rate f will be explained in more detail in the following sections.

The motion of liquid alumina concentration can be described by a convection-diffusion equation,

$$\frac{\partial c(\vec{x},t)}{\partial t} + \vec{u}(\vec{x}).\vec{\nabla} c(\vec{x},t) - D\Delta c(\vec{x},t) = \dot{Q}(\vec{x},t). \tag{1.2.3}$$

Here, D is the diffusion coefficient of alumina in the bath and $\dot{Q}(\vec{x},t) = \dot{q}_1(\vec{x},t) + \dot{q}_2(\vec{x},t)$ is the source of liquid alumina which consists of two parts. On one hand, we have the creation of concentration due to the dissolution of the particles, which we will denote by $\dot{q}_1$. On the other hand, we have the consumption of concentration because of the electrolysis, denoted by $\dot{q}_2$. We will now describe the different parts appearing in the equations.

Feeding of alumina particles

We consider first the source term S of the convection equation for the alumina particles (1.2.1), which is due to the feeding of the particles. The function S can be written as a product of three functions, one for each variable:

$$S(\vec{x},R,t) = C_S S_1(\vec{x}) S_2(R) S_3(t).$$

The total mass added to the bath at each feeding is adjusted by the constant C_S. We will first describe the functions S_1, S_2 and S_3 and then define C_S.

The feeding takes place at specified times $\{\tau^i\}$, and thus the feeding function in time is a sum of Dirac-functions. Hence, S_3 can be written as

$$S_3(t) = \sum_i \delta_{\tau^i}(t).$$

The function S_2 is the particle size distribution of the alumina powder. The aluminium industry has requirements on the mass distribution as a function of the particle size, but these requirements only specify the mass percentage for certain discrete particle sizes and not as a continuous curve. The data is given as a cumulative distribution function, i.e., for each data point $(s_i, y_i), i = 1, \dots, k$, s_i is the particle diameter and y_i is the mass of all the particles of diameter smaller than s_i. The total mass of the particles is normalized to 1. In this form, the discrete distribution can be approximated by a continuous probability distribution. For the approximation we take the cumulative log-normal distribution function, which is given by

$$y = g(s;\mu,\sigma) = \frac{1}{2} + \frac{1}{\sqrt{\pi}} \int_0^{\frac{\log s - \mu}{\sqrt{2\sigma^2}}} e^{-t^2}\,dt, \quad s > 0, \tag{1.2.4}$$

where $\mu \in \mathbb{R}$ and $\sigma > 0$ are parameters (see [11] for more details). The parameters are determined by a nonlinear fit in the following way. Given k data points, where each data point is a couple of values (s_i, y_i), the nonlinear fit finds values of the parameters μ and σ such that

$$\sum_{n=1}^{k} \left(g(s_n;\mu,\sigma) - y_n\right)^2 \tag{1.2.5}$$

is minimized. We compute this nonlinear fit using the program Maple$^{\text{TM}}$ [27]. The mass density function $(4/3)\pi\rho s^3 S_2(s)$ is then equal to the log-normal density function with the parameters found, which is given by

$$\frac{4}{3}\pi\rho s^3 S_2(s) = \frac{1}{s\sigma\sqrt{2\pi}} \exp\left(-\frac{(\log s - \mu)^2}{2\sigma^2}\right), \quad s > 0. \tag{1.2.6}$$

The relation between the cumulative log-normal distribution function g and the function S_2 is

$$g(s;\mu,\sigma) = \frac{4}{3}\pi\rho \int_0^s \tau^3 S_2(\tau) d\tau. \tag{1.2.7}$$

In Figure 1.1 we show the data (from [36]) in the form of a cumulative distribution function, as well as the cumulative distribution g, both as functions of the particle diameter $2R$. For g the values of the parameters are fixed to $\mu = -0.31$ and $\sigma = 0.38$.

Alumina is added under the channel between the anodes. We suppose that S_1 is of the form (with $\vec{x} = (x, y, z)$)

$$S_1(\vec{x}) = \exp\left(-\frac{1}{1-(x-x_0)^2/a_1^2 - y^2/a_2^2 - (z-z_0)^2/a_3^2}\right) \tag{1.2.8}$$

if the denominator of the fraction is greater than zero, and $S_1(\vec{x}) = 0$ otherwise. Thus S_1 is nonzero in an ellipsoid centred at $(x_0, 0, z_0)$ and with semi-principal axes of length (a_1, a_2, a_3).

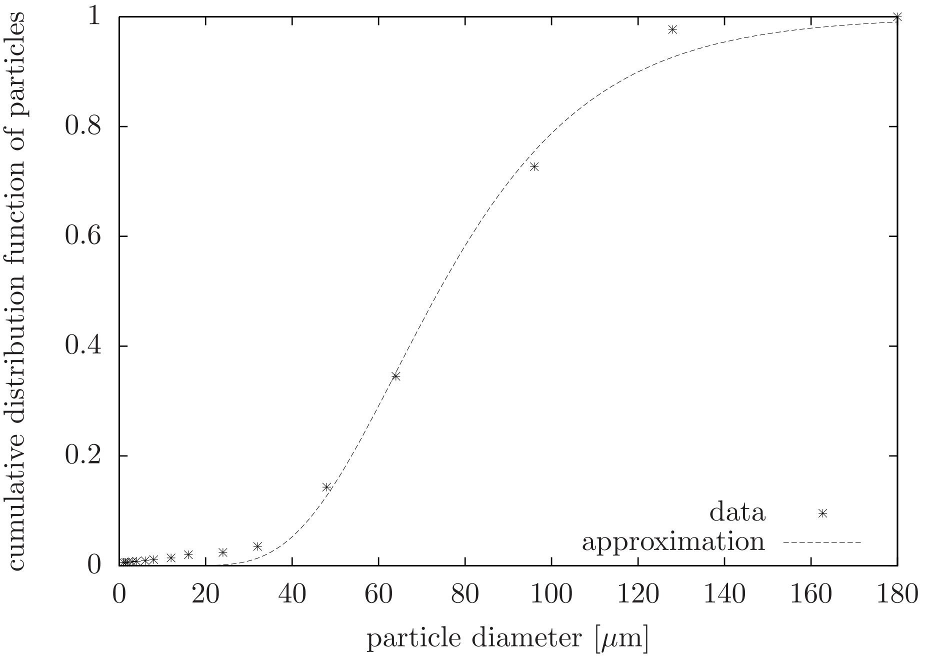


Figure 1.1: The initial mass distribution of the alumina particles with respect to the particle diameter, data [36] and approximated curve.

The mass of alumina which is added at each time τ^j is equal to a constant C_m, which depends on the consumption of liquid alumina concentration by electrolysis, which in turn depends on the current passing through the bath. We have

$$C_m = \frac{IM}{6F}, \tag{1.2.9}$$

where I is the electric current, M the molar mass of alumina, and F the Faraday constant. In order to add the right amount of alumina we compute

$$C'_m = \int_{\Omega_{\text{el}}} S_1(\vec{x}) d\Omega \tag{1.2.10}$$

and we set

$$C_S = \frac{C_m}{C'_m}. \tag{1.2.11}$$

In this way the mass added is equal to C_m. We note that the mass contribution of $S_2(R)$ is equal to 1 because it is issue of a probability density function.

Dissolution process

The dissolution kinetics is supposed to be the chemical reaction, but since the reactions are not precisely known, and since they are hard to compute, we take a measured dissolution curve [32] to simulate the dissolution. The curve specifies the time t of complete dissolution as a function of the particle size R. The measure was executed in a big recipient to avoid that the appearing concentration saturated the bath and thus slowed down the dissolution process. Hence, we obtained the curve

$$t = h(R), \tag{1.2.12}$$

which is shown on Figure 1.2 and which does not depend on the concentration. The particles are always getting smaller, i.e., $h'(t) > 0, \forall t$, and thus the function h is bijective and invertible with inverse function $h^{-1}(t) = R(t)$. We know that

$$\frac{dR(t)}{dt} = \left(h^{-1}\right)'(t) = \frac{1}{h'(h^{-1}(t))} = \frac{1}{h'(R(t))}. \tag{1.2.13}$$

By the definition of f, for each $\vec{x} \in \Omega_{\text{el}}$ we have

$$f(R(t), c(\vec{x}, t)) = \frac{dR(t)}{dt} = \frac{1}{h'(R(t))}. \tag{1.2.14}$$

Since the dissolution process depends on the concentration, at least if the concentration is close to saturation, we multiply the function f by the factor $(c_{\text{sat}} - c(\vec{x}, t))$, where c_{sat} is the saturation concentration. With this factor we assure that the concentration will not go beyond saturation.

To give a more concrete example of the dissolution, we can take f of the form

$$f(R, c) = -\frac{DM(c_{\text{sat}} - c)}{\rho R}, \tag{1.2.15}$$

where D is the diffusion coefficient of alumina in the bath, M is the molar mass of alumina, and ρ is its mass density. This is the dissolution law if the dissolution kinetics is diffusion. In this case the particle radius diminishes faster as the particle gets smaller. For a derivation of the model, see for example Haverkamp and Welch [22] or Wang and Flanagan [37]. We will in general work with the measured dissolution curve, but the details of the discretization of the diffusion law will be given in Section 4.1.

Source terms of the concentration

The concentration which appears due to the dissolution of the particles is given by

$$\dot{q}_1(\vec{x}, t) = -4\pi \frac{\rho}{M} \int_0^\infty n_p(\vec{x}, R, t) f(R, c(\vec{x}, t)) R^2 dR. \tag{1.2.16}$$

At the same time, concentration diminishes by the electrolytic process. This is supposed to happen with uniform intensity in the part of the bath which lies between the bottom of the anodes and the interface bath - aluminium. Since the transformation of Al_2O_3 to Al requires 6 electrons, the consumption of the concentration is given by

$$\dot{q}_2 = -\frac{I}{6FV}, \tag{1.2.17}$$

where I is the total electric current, F the Faraday constant, and V the part of the volume of the bath Ω_{el} which lies between the bottom of the anodes and the interface bath - aluminium.

Boundary conditions

The velocity field $\vec{u}$ is supposed to verify either $\vec{u}.\vec{n} = 0$ or $\vec{u} = 0$ on the boundary of the bath, denoted by $\partial\Omega_{\text{el}}$. Therefore, equation (1.2.1) does not need any boundary conditions.
For the convection-diffusion equation of the concentration we want to avoid outflow and inflow at the boundaries and we therefore impose

$$\frac{\partial c}{\partial n} = 0, \text{ on } \partial\Omega_{\text{el}} \times (0, T).$$

Complete mathematical model

The complete mathematical model is thus:

Problem 1.2.1. Find $c = c(\vec{x}, t)$ and $n_p = n_p(\vec{x}, R, t)$ verifying

$$\frac{\partial c}{\partial t} + \vec{u}.\vec{\nabla}c - D\Delta c = \dot{q}_1 + \dot{q}_2 = \dot{Q}, \qquad \text{in } \Omega_{\text{el}} \times (0, T), \tag{1.2.18}$$

$$\frac{\partial c}{\partial n} = 0, \qquad \text{on } \partial\Omega_{\text{el}} \times (0, T), \tag{1.2.19}$$

$$\frac{\partial n_p}{\partial t} + \vec{u}.\vec{\nabla}_x n_p + \frac{\partial}{\partial R}(n_p f(\cdot, c)) = S, \qquad \text{in } \Omega_{\text{el}} \times (0, \infty) \times (0, T), \tag{1.2.20}$$

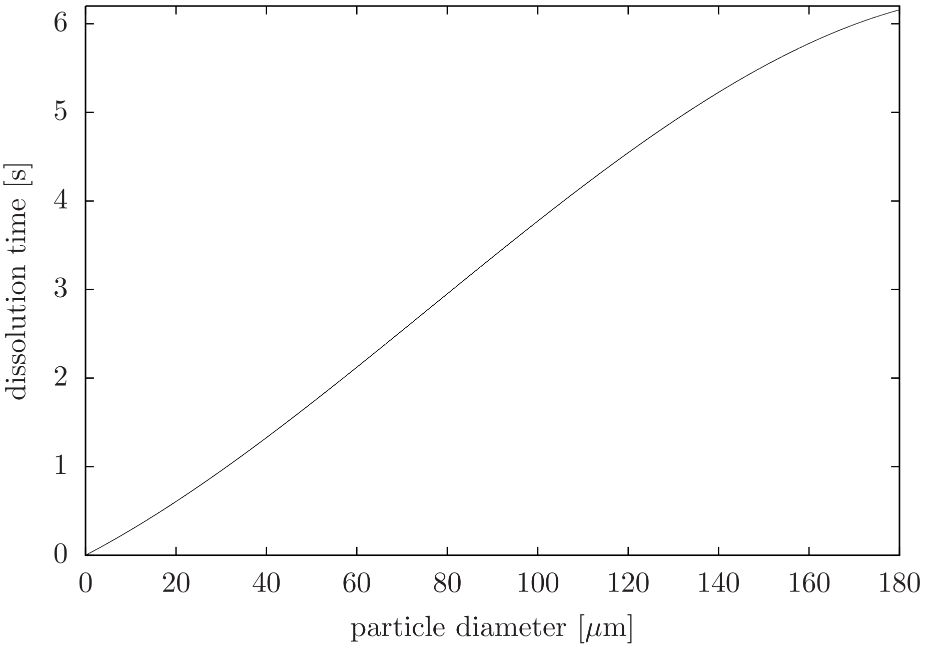


Figure 1.2: The dissolution time t for alumina particles as a function of the particle diameter $2R$. Curve received from [32].

with initial conditions $c(\vec{x}, t = 0) = c^0(\vec{x})$ and $n_p(\vec{x}, R, t = 0) = n_p^0(\vec{x}, R)$, and where

$$\dot{Q}(\vec{x}, t) = -4\pi \frac{\rho}{M} \int_0^\infty n_p(\vec{x}, R, t) f(R, c(\vec{x}, t)) R^2 dR - \frac{I}{6FV}, \tag{1.2.21}$$

and

$$S((x, y, z), R, t) = C_S \exp\left(-\frac{1}{1 - (x - x_0)^2/a_1^2 - y^2/a_2^2 - (z - z_0)^2/a_3^2}\right)$$
$$\frac{3}{4\sqrt{2}(2R)^4 \sigma\rho\pi^{3/2}} \exp\left(-\frac{(\log(2R) - \mu)^2}{2\sigma^2}\right) \left(\sum_i \delta_{\tau^i}(t)\right). \tag{1.2.22}$$

The parameters $C_S, a_1, a_2, a_3, x_0, z_0, \mu, \sigma, \tau^i$ are explained on pages 11-13. The particle dissolution function f is described on page 13.

Properties of the model

We consider first the convection equation for the particles without source term:

$$\frac{\partial n_p(\vec{x}, R, t)}{\partial t} + \vec{u}(\vec{x}).\vec{\nabla}_x n_p(\vec{x}, R, t) + \frac{\partial}{\partial R}(n_p(\vec{x}, R, t) f(R, c(\vec{x}, t))) = 0. \tag{1.2.23}$$

Over time, this equation does not change the number of particles, since $\operatorname{div} \vec{u} = 0$ and $\vec{u}.\vec{n} = 0$ on the boundary $\partial\Omega_{\text{el}}$ of the bath Ω_{el}, but it does diminish the total mass of the particles because particle radii become smaller by dissolution. We show first that the number of particles remains constant. The number of particles is computed by integrating n_p with respect to space and radius. Thus, integrating the two last terms of equation (1.2.23) with respect to $\vec{x}$ and to R we get

$$\int_0^\infty \int_{\Omega_{\text{el}}} \vec{u}.\vec{\nabla}_x n_p d\Omega dR = \int_0^\infty \int_{\Omega_{\text{el}}} (\operatorname{div}(\vec{u} n_p) - n_p \operatorname{div} \vec{u})\, d\Omega dR = \int_0^\infty \int_{\partial\Omega_{\text{el}}} n_p \vec{u}.\vec{n} d\Omega dR = 0, \tag{1.2.24}$$

$$\int_{\Omega_{\text{el}}} \int_0^\infty \frac{\partial}{\partial R}(n_p f) dR d\Omega = \int_{\Omega_{\text{el}}} \left(\lim_{R\to\infty} (n_p f) - (n_p f)|_{R=0}\right) d\Omega = 0. \tag{1.2.25}$$

Here we used the facts that particles of radius zero do not dissolve, i.e., $f(0, c) = 0$, and that the particles have finite size, i.e., $\lim_{R\to\infty} n_p(\vec{x}, R, t) = 0$. Integrating now the first term of equation (1.2.23) and using the two previous equations we obtain

$$\frac{d}{dt} \int_{\Omega_{\text{el}}} \int_0^\infty n_p dR d\Omega = 0. \tag{1.2.26}$$

Hence, the number of particles is conserved.

The mass of the particles is given by

$$\frac{4}{3}\pi\rho \int_{\Omega_{\text{el}}} \int_0^\infty n_p R^3 dR d\Omega.$$

Therefore, to show the decrease of the mass for increasing time t we multiply equation (1.2.23) by R^3 before integrating. The integration of the convection term is analogous to the computation of (1.2.24). Integrating the term with the derivative in R we get

$$\int_{\Omega_{\text{el}}} \int_0^\infty \frac{\partial(n_p f)}{\partial R} R^3 dR d\Omega = -\int_{\Omega_{\text{el}}} \int_0^\infty 3R^2 n_p f dR d\Omega. \tag{1.2.27}$$

We have $R \geq 0$, $n_p \geq 0$ and, since the particles get smaller, $f < 0$. Hence, we find that

$$\int_{\Omega_{\text{el}}} \int_0^\infty \frac{\partial n_p}{\partial t} R^3 dR d\Omega = \int_{\Omega_{\text{el}}} \int_0^\infty 3R^2 n_p f dR d\Omega < 0, \tag{1.2.28}$$

thus, the total mass of the particles decreases in time.

We now look at the mass in the complete model. We want to show that without the feeding of alumina particles (i.e. $S = 0$ in (1.2.20)) and without electrolysis (i.e. $\dot{q}_2 = 0$ in (1.2.18)) the total mass of alumina in the bath does not change over time. The change of the total mass of alumina is obtained by multiplying equation (1.2.18) by M and integrating over space, by multiplying equation (1.2.20) by $4/3\pi\rho R^3$ and integrating over space and radius, and finally taking the sum of the two equations. The contribution of (1.2.18) is

$$\frac{d}{dt} \int_{\Omega_{\text{el}}} M c d\Omega = -\int_{\Omega_{\text{el}}} 4\pi\rho \int_0^\infty n_p f R^2 dR d\Omega, \tag{1.2.29}$$

(the convection and the diffusion term do not contribute to the change of the total mass in the bath). From equation (1.2.20) we get

$$\frac{4}{3}\pi\rho \frac{d}{dt} \int_{\Omega_{\text{el}}} \int_0^\infty n_p R^3 dR d\Omega = -\frac{4}{3}\pi\rho \int_{\Omega_{\text{el}}} \int_0^\infty \frac{\partial(n_p f)}{\partial R} R^3 dR = 4\pi\rho \int_{\Omega_{\text{el}}} \int_0^\infty R^2 n_p f dR d\Omega. \tag{1.2.30}$$

Adding the two last equations leads to

$$\frac{d}{dt}\left(M \int_{\Omega_{\text{el}}} c d\Omega + \frac{4}{3}\pi\rho \int_{\Omega_{\text{el}}} \left(\int_0^\infty n_p R^3 dR \right) d\Omega \right) = 0, \tag{1.2.31}$$

and thus the total mass of alumina does not change over time. This shows that our model verifies an important conservation law.

1.3 Computation of the velocity field

We present here briefly how the numerical approximation of the velocity field $\vec{u}_h$, which we use in our model, has been computed. The following is a summary of [13].

The velocity field is computed in the whole domain of the liquids Ω, i.e., in the domain of the bath Ω_{el} and in the domain of the liquid aluminium Ω_{al}. To find $\vec{u}$ and the pressure p, the steady-state Navier-Stokes equations are solved in Ω. At the same time, the interface Γ between the two liquids has to be determined.

A quantity v is denoted by v^1 if we refer to the part defined in Ω_{el} and by v^2 if we refer to the part defined in Ω_{al}. We denote by $[v]_\Gamma = v^1 - v^2$ the jump of v through the interface Γ.

Some conditions appear at the interface Γ between the two liquids. The liquids are immiscible, which at steady-state gives the condition

$$\vec{u}.\vec{n} = 0, \quad \text{on } \Gamma, \tag{1.3.1}$$

where $\vec{n}$ is the unit normal of Γ pointing upwards. Then, the liquids have to satisfy no-slip conditions on the interface,

$$[\vec{u}]_\Gamma = 0. \tag{1.3.2}$$

And finally, the interface has to be in a mechanical equilibrium position, i.e.,

$$[\sigma\vec{n}]_\Gamma = \vec{0}, \tag{1.3.3}$$

where $\sigma_{ij}(\vec{u},p) = -p\delta_{ij} + 2\eta\epsilon_{ij}(\vec{u})$. The velocity gradient is $\epsilon_{ij}(\vec{u}) = \frac{1}{2}\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right)$ and δ_{ij} is the Kronecker symbol. For our computations we have two possible viscosity tensors η. It is either a laminar viscosity tensor, given by

$$\eta = \begin{pmatrix} \alpha_\eta & \alpha_\eta & \beta_\eta \\ \alpha_\eta & \alpha_\eta & \beta_\eta \\ \beta_\eta & \beta_\eta & \gamma_\eta \end{pmatrix}, \tag{1.3.4}$$

with $\alpha_\eta = 10$ [Pa s], $\beta_\eta = 0.5$ [Pa s] and $\gamma_\eta = 1$ [Pa s]. This particular form of the viscosity tensor was chosen for geometric reasons (the horizontal dimensions of Ω are much larger than the height of Ω), and the actual values were validated with real experiments by Romerio [31]. Or, using a different model, it can be a turbulent viscosity tensor, of the form

$$\eta_{ij} = \eta_l + \alpha\rho\sqrt{2\epsilon(\vec{u}) : \epsilon(\vec{u})}, \tag{1.3.5}$$

where η_l is a laminar tensor with $\eta_{ij,l} = 0.002$ [Pa s] and α is a numerical coefficient which has been validated to be in the order 10^{-4} [m^2] [34].

The computation of the velocity field is done as an iteration over two stages. In the first stage the Navier-Stokes problem is solved with a fixed interface, and in the second stage the interface is corrected to adjust for the jump of the normal stresses. The two stages are iteratively solved until the jump of the normal stresses is constant on Γ. This algorithm is called interface algorithm since it aims at computing the steady-state interface between the bath and the metal pad. Once this algorithm has converged we have the steady-state velocity field which verifies equations (1.3.1) - (1.3.3).

The first stage consists thus in solving the following problem:

Problem 1.3.1. For Γ fixed, compute the body force $\vec{f}$ which is the sum of electromagnetic forces and gravity forces (see [34]). Then find $\vec{u} : \Omega \to \mathbb{R}^3$ and $p : \Omega \to \mathbb{R}$ such that

$$\begin{aligned}
&\operatorname{div}\sigma + \vec{f} = \rho(\vec{u}.\vec{\nabla})\vec{u}, && \text{in } \Omega_{\text{el}} \cup \Omega_{\text{al}}, && (1.3.6)\\
&\operatorname{div}\vec{u} = 0, && \text{in } \Omega_{\text{el}} \cup \Omega_{\text{al}}, && (1.3.7)\\
&\vec{u} = 0, && \text{on } \partial\Omega, && (1.3.8)\\
&\vec{u}.\vec{n} = 0, && \text{on } \Gamma, && (1.3.9)\\
&[\vec{u}]_\Gamma = 0, && && (1.3.10)\\
&[\vec{n}^T\sigma\vec{t}]_\Gamma = 0, && \forall\vec{t} \text{ such that } \vec{t}.\vec{n} = 0. && (1.3.11)
\end{aligned}$$

The pressure p can have a jump across Γ.

In the second stage the parametrisation of the interface Γ, which is given by $(x,y) \mapsto (x,y,h(x,y))$, is corrected by changing h to $h + \delta h$, with

$$\delta h = -\frac{\psi + C}{[f_3]_\Gamma}. \tag{1.3.12}$$

Here the constant C ensures volume conservation of Ω_{el}, $\psi = [\vec{n}^T\sigma\vec{n}]_\Gamma$ is the jump of the normal stresses, and f_3 is the third component of $\vec{f}$.

We set $W = (H_0^1(\Omega))^3$, $Q = L_0^2(\Omega)$, and $Y = L^2(\Gamma)$, where $H_0^1(\Omega) = \{u \in L^2(\Omega) : \partial u/\partial x_i \in L^2(\Omega), i = 1,2,3, u = 0 \text{ on } \partial\Omega\}$, and $L_0^2(\Omega) = \{v \in L^2(\Omega) | \int_\Omega v d\Omega = 0\}$. The weak formulation of Problem 1.3.1 can be written as:

Problem 1.3.2. Find $\vec{u} \in W$, $p \in Q$ and $\psi \in Y$ such that

$$\sum_{i,j=1}^{3} \int_\Omega \sigma_{ij}\epsilon_{ij}(\vec{v})d\Omega + \sum_{i,j=1}^{3} \int_\Omega \rho u_j \frac{\partial u_i}{\partial x_j} v_i d\Omega + \int_\Gamma \psi(\vec{v}.\vec{n})d\sigma = \int_\Omega \vec{f}.\vec{v}d\Omega, \quad \forall \vec{v} \in W, \tag{1.3.13}$$

$$\int_\Omega q \operatorname{div} \vec{u} d\Omega = 0, \quad \forall q \in Q, \tag{1.3.14}$$

$$\int_\Gamma \xi \vec{u}.\vec{n} d\sigma = 0, \quad \forall \xi \in Y. \tag{1.3.15}$$

Two different discretizations of Problem 1.3.2 are used. In the first, a Galerkin least-squares (GLS) stabilization which was proposed by Franca and Hughes [16] is applied. Let $h > 0$ and let $\mathcal{T}_h$ be a tetrahedral mesh of Ω, with all the tetrahedrons K of $\mathcal{T}_h$ of diameter smaller than h. Defining the spaces W_h, Q_h and Y_h as P_1-approximations of W, Q and Y, the discretized problem is:

Problem 1.3.3. Starting with $\vec{w}_h = 0$, we find $\vec{u}_h \in W_h$, $p_h \in Q_h$ and $\psi_h \in Y_h$ such that $\forall \vec{v}_h \in W_h$, $\forall q_h \in Q_h$ and $\forall \xi_h \in Y_h$

$$\begin{aligned} &\sum_{i,j=1}^{3} \int_\Omega \sigma_{h,ij}\epsilon_{ij}(\vec{v}_h)d\Omega + \sum_{i,j=1}^{3} \int_\Omega \rho w_{h,j} \frac{\partial u_{h,i}}{\partial x_j} v_{h,i} d\Omega + \int_\Gamma \psi_h(\vec{v}_h.\vec{n})d\sigma \\ &\qquad + \sum_{K \in \mathcal{T}_h} \int_K \vec{\nu}_1 \rho \left(\rho(\vec{w}_h.\vec{\nabla})\vec{u}_h - \operatorname{div}\sigma_h - f_h\right)(\vec{v}_h.\vec{\nabla})\vec{v}_h d\Omega = \int_\Omega \vec{f}_h.\vec{v}_h d\Omega, \end{aligned} \tag{1.3.16}$$

$$\int_\Omega q_h \operatorname{div} \vec{u}_h d\Omega + \sum_{K \in \mathcal{T}_h} \int_K \nu_2 \left(\rho(\vec{w}_h.\vec{\nabla})\vec{u}_h - \operatorname{div}\sigma_h - f_h\right)\vec{\nabla} q_h d\Omega = 0, \tag{1.3.17}$$

$$\int_\Gamma \xi_h(\vec{u}_h.\vec{n})d\sigma = 0, \tag{1.3.18}$$

where, setting $\mu_\eta = \min\{\alpha_\eta, \beta_\eta, \gamma_\eta\} = 0.5$,

$$\vec{\nu}_1 = \begin{cases} \frac{h_K}{12\rho u_K}(1,1,1), & \text{if } \frac{\rho u_K h_K}{\mu_\eta} \geq 1, \\ \frac{h_K^2}{12\mu_\eta}(1,1,1), & \text{otherwise}, \end{cases} \qquad \nu_2 = \begin{cases} 2\rho h_K u_K, & \text{if } \frac{\rho u_K h_K}{\mu_\eta} \geq 1, \\ 6\mu_\eta, & \text{otherwise}, \end{cases} \tag{1.3.19}$$

with u_K the mean of the Euclidean norm of $\vec{w}_h$ on K. We then iterate by Picard's method, setting $\vec{w}_h = \vec{u}_h$ and solving the problem again until convergence of $\vec{u}_h$ to $\vec{w}_h$.

Once we obtain the velocity field, the interface is corrected according to (1.3.12). This correction modifies the mesh by slightly moving the nodes of the interface. The body force $\vec{f}_h$ changes because the interface position is modified, it is therefore recomputed and Problem 1.3.3 is solved on the new mesh and with the new force term. This iteration is executed until convergence of the interface.

The velocity field $\vec{u}_h$ we obtain by solving this problem is in general (i.e., for nontrivial $\vec{u}_h$ and for nontrivial $q_h \in Q_h$) such that

$$\int_{\Omega_{\text{el}}} q_h \operatorname{div} \vec{u}_h \neq 0, \quad q_h \in Q_h, \tag{1.3.20}$$

because of the stabilization terms.

In the second approximation we take again a tetrahedral mesh $\mathcal{T}_h$ as before, we define the space of bubble functions

$$B_h = \{\vec{v}_h : \vec{v}_h|_K \in (P_4(K))^3 \cap (H_0^1(K))^3, \forall K \in \mathcal{T}_h\}, \tag{1.3.21}$$

and we set $X_h = W_h \oplus B_h$. This was proposed by Arnold, Brezzi and Fortin [3]. The discretization of Problem 1.3.2 becomes

Problem 1.3.4. Find $\vec{u}_h \in X_h$, $p_h \in Q_h$ and $\psi_h \in Y_h$ such that $\forall \vec{v}_h \in X_h$, $\forall q_h \in Q_h$ and $\forall \xi_h \in Y_h$

$$\sum_{i,j=1}^{3} \int_\Omega \sigma_{h,ij}\epsilon_{ij}(\vec{v}_h) d\Omega + \sum_{i,j=1}^{3} \int_\Omega \rho u_{h,i} \frac{\partial u_{h,i}}{\partial x_j} v_{h,j} d\Omega + \int_\Gamma \psi_h(\vec{v}_h.\vec{n}) d\sigma = \int_\Omega \vec{f}_h.\vec{v}_h d\Omega, \tag{1.3.22}$$

$$\int_\Omega q_h \operatorname{div} \vec{u}_h d\Omega = 0, \tag{1.3.23}$$

$$\int_\Gamma \xi_h(\vec{u}_h.\vec{n}) d\sigma = 0. \tag{1.3.24}$$

We solve this problem using Newton's method on the nonlinear term. As before, we then correct the interface, giving a new mesh, and we compute the new force term and solve again Problem 1.3.4. We iterate those two stages until the interface converges.

By solving this problem we find a velocity field which verifies

$$\int_{\Omega_{\text{el}}} q_h \operatorname{div} \vec{u}_h d\Omega = 0, \quad \forall q_h \in Q_h. \tag{1.3.25}$$

Remark 1.3.5. In case we need to distinguish between the two velocity fields we will denote the one obtained using the GLS discretization by $\vec{u}_h^{\text{GLS}}$ and the one obtained using the bubble discretization by $\vec{u}_h^{\text{bubble}}$. We note that assertion (1.3.25) is only true if $\vec{u}_h^{\text{bubble}}$ is taken in X_h, i.e., if we keep the bubble functions in $\vec{u}_h^{\text{bubble}}$. If we only take $\vec{u}_h^{\text{bubble}} \in W_h$, implying that $\vec{u}_h^{\text{bubble}}$ is linear on each tetrahedron, the velocity field is not better than $\vec{u}_h^{\text{GLS}}$ when considering $\int_{\Omega_{\text{el}}} q_h \operatorname{div} \vec{u}_h d\Omega$.

Remark 1.3.6. When solving the discretized Problems 1.3.2 and 1.3.4, the solutions do not exactly verify the equations because the computations are not exact on a computer. Thus, even for $\vec{u}_h^{\text{bubble}}$ the crucial integrals $\int_\Omega q_h \operatorname{div} \vec{u}_h d\Omega$ and $\int_\Gamma \xi_h(\vec{u}_h.\vec{n}) d\sigma$ will not exactly vanish. However, this error might be negligible.

Remark 1.3.7. It is also possible to solve the unsteady Navier-Stokes equations in order to get the velocity field, as is done in [34].

Chapter 2

Numerical approximation of a convection-diffusion equation

The model presented in the preceding chapter contains a convection and a convection-diffusion equation, both of them using the same approximation of a velocity field. In the present chapter we will study a model convection-diffusion equation, using a velocity field approximated in the same manner as for the alumina problem. We will first describe the problem and state the properties which the approximated solution should have. We then proceed by giving several finite element approximations of the convection-diffusion equation. For each scheme, we will check if its solution shows the desired properties. We conclude this chapter by choosing the scheme that we will use subsequently.

2.1 Model problem and desired properties of the solution

In this chapter we want to find the solution of the following convection-diffusion problem:

Problem 2.1.1. Find $c = c(\vec{x}, t)$ verifying

$$\frac{\partial c}{\partial t} - D\Delta c + \vec{u}.\vec{\nabla} c = f, \qquad \text{in } \Omega \times (0, T), \tag{2.1.1}$$

$$\frac{\partial c}{\partial n} = 0, \qquad \text{on } \partial\Omega \times (0, T), \tag{2.1.2}$$

$$c = c^0, \qquad \text{in } \Omega \times \{0\}. \tag{2.1.3}$$

We assume $\Omega \subset \mathbb{R}^3$, the velocity $\vec{u} = \vec{u}(\vec{x})$ is not depending on time and verifies

$$\text{div}(\vec{u}) = 0 \text{ in } \Omega \text{ and } \vec{u}.\vec{n} = 0 \text{ on } \partial\Omega, \tag{2.1.4}$$

where $\vec{n}$ is the external unit normal of Ω. The final time is denoted by T.
We want to compute an approximation c_h of c which has the following three properties:

- Mass conservation
$$\frac{d}{dt}\int_\Omega c_h d\Omega = \int_\Omega f d\Omega; \tag{2.1.5}$$
- Conservation of the trivial solution
$$\text{if } f = 0 \text{ and } c^0 = \text{constant}, \text{then } c_h = c^0; \tag{2.1.6}$$

- L^2-stability

$$\frac{d}{dt}\|c_h(t)\|_{L^2(\Omega)} \leq \|f(t)\|_{L^2(\Omega)}, \forall t > 0. \tag{2.1.7}$$

It is reasonable to ask for these properties, because the solution to Problem 2.1.1, with a velocity field $\vec{u}$ that verifies conditions (2.1.4), shows them all.
Note that the first property is called "mass conservation", because if $f = 0$ it states that the mass of c_h in Ω is conserved.
When computing c_h we will not work with the original velocity field $\vec{u}$ but with some computed approximation $\vec{u}_h$. This approximation will in general not verify conditions (2.1.4), maybe because of the way $\vec{u}_h$ was approximated, but also because of small numerical errors. These small errors make it hard to attain the goals which we just stated.
For this reason, we will in the following sections present different semi-discretized weak formulations of Problem 2.1.1. For every formulation we will check whether the solution shows mass conservation, whether the trivial solution is a solution of the problem and whether the solution is L^2-stable. If there are any particularities related to a method we will also state them. We will do this examination assuming that the velocity field $\vec{u}_h$ is such that $\operatorname{div}\vec{u}_h$ in Ω and $\vec{u}_h.\vec{n}$ on $\partial\Omega$ are not exactly vanishing.
For the semidiscretization in space let $h > 0$ and let $\mathcal{T}_h$ be a tetrahedral mesh of Ω, where all the tetrahedrons K of the mesh are of diameter $\operatorname{diam}(K) \leq h$. We define the finite element space V_h of piecewise polynomial functions $P_1(K)$ of degree 1 on K by

$$V_h = \{g \in C^0(\Omega) : g|_K \in P_1(K), \forall K \in \mathcal{T}_h\}. \tag{2.1.8}$$

The initial condition for all schemes is $c_h(0) = c^0$ if c^0 is given in V_h, and if this is not the case we take a projection $c_h^0 \in V_h$ of c_0 as initial condition $c_h(0)$.
We will use standard notations for the spaces $H^1(0,T;V_h), C^1([0,T];H^2(\Omega)), \ldots$, which are defined in [12].
A numerical comparison of the solutions of the different formulations will be done in Section 5.2.

2.2 Direct weak formulation

A standard finite element approximation in space for computing an approximation c_h of c is to look for $c_h \in H^1(0,T;V_h)$ such that for all $v \in V_h$

$$\int_\Omega \frac{\partial c_h}{\partial t} v d\Omega + D\int_\Omega \vec{\nabla} c_h \vec{\nabla} v d\Omega + \int_\Omega \vec{u}_h.\vec{\nabla} c_h v d\Omega = \int_\Omega f v d\Omega. \tag{2.2.1}$$

(see [30]). We check the three conditions stated above:

Mass conservation. We take $v \equiv 1$. Then (2.2.1) simplifies to

$$\frac{d}{dt}\int_\Omega c_h d\Omega + \int_\Omega \vec{u}_h.\vec{\nabla} c_h d\Omega = \int_\Omega f d\Omega. \tag{2.2.2}$$

But

$$\int_\Omega \vec{u}_h.\vec{\nabla} c_h d\Omega = \int_{\partial\Omega} c_h \vec{u}_h.\vec{n} d\sigma - \int_\Omega \operatorname{div}(\vec{u}_h) c_h d\Omega, \tag{2.2.3}$$

and the right hand side does generally not vanish under our assumptions, hence mass conservation cannot be guaranteed.

Conservation of trivial solution. If we suppose that $f = 0$ and $c^0 = 1$ then $c_h(\vec{x}, t) = c^0(\vec{x})$ is the unique solution of (2.2.1).

L^2-stability. We take $v = c_h$. Then (2.2.1) becomes

$$\frac{1}{2}\frac{d}{dt}\int_\Omega c_h^2 d\Omega + D\int_\Omega |\vec{\nabla} c_h|^2 d\Omega = \int_\Omega f c_h d\Omega - \frac{1}{2}\int_\Omega \vec{u}_h.\vec{\nabla}(c_h^2) d\Omega. \tag{2.2.4}$$

Since

$$\int_\Omega \vec{u}_h.\vec{\nabla}(c_h^2) d\Omega = \int_{\partial\Omega} c_h^2 \vec{u}_h.\vec{n} d\sigma - \int_\Omega \operatorname{div}(\vec{u}_h) c_h^2 d\Omega \tag{2.2.5}$$

is generally not vanishing we can not ensure the L^2-stability.

Remark 2.2.1. In the above examination we supposed that $\vec{u}_h$ does not verify both of the conditions of equation (2.1.4). When we presented the computation of $\vec{u}_h$ in Section 1.3 we pointed out that for $\vec{u}_h^{\text{bubble}}$ we have

$$\int_\Omega q_h \operatorname{div} \vec{u}_h^{\text{bubble}} d\Omega = 0, \quad \forall q_h \in Q_h, \tag{2.2.6}$$

$$\int_\Gamma \xi_h \vec{u}_h^{\text{bubble}}.\vec{n} d\sigma = 0, \quad \forall \xi_h \in Y_h. \tag{2.2.7}$$

If we apply these two results to our model problem this implies that

$$\int_\Omega c_h \operatorname{div} \vec{u}_h^{\text{bubble}} d\Omega = 0, \qquad \int_\Gamma c_h \vec{u}_h^{\text{bubble}}.\vec{n} d\sigma = 0, \tag{2.2.8}$$

hence, in (2.2.3) we have equality and thus the mass is conserved. This is correct up to the tolerance which was chosen when solving the linear system arising from problem (1.3.4). We also note that we still do not get L^2-stability, because in (2.2.5) we have the factor c_h^2. This is not a linear function, and thus it is not in the abovementioned spaces Q_h and Y_h.

2.3 Simple mass conserving formulation

We take the formulation from the preceding section, but now we integrate the velocity integral by parts. The new formulation is thus to find $c_h \in H^1(0, T; V_h)$ such that for all $v \in V_h$

$$\int_\Omega \frac{\partial c_h}{\partial t} v d\Omega + D\int_\Omega \vec{\nabla} c_h \vec{\nabla} v d\Omega - \int_\Omega \vec{u}_h.\vec{\nabla} v c_h d\Omega = \int_\Omega f v d\Omega. \tag{2.3.1}$$

We check the three conditions:

Mass conservation. Setting $v \equiv 1$ in (2.3.1) we get

$$\frac{d}{dt}\int_\Omega c_h d\Omega = \int_\Omega f d\Omega, \tag{2.3.2}$$

which is exactly what we want and justifies the title of the section.

Conservation of trivial solution. If $f = 0$ and $c^0 = 1$ then

$$\int_\Omega \frac{\partial c_h}{\partial t} d\Omega - \int_\Omega \vec{u}_h.v d\Omega = 0. \tag{2.3.3}$$

This velocity integral will in general not vanish and thus the solution will be time-dependent.

L^2-stability. We take $v = c_h$. Equation (2.3.1) becomes

$$\frac{1}{2}\frac{d}{dt}\int_\Omega c_h^2 d\Omega + D\int_\Omega |\vec{\nabla} c_h|^2 d\Omega = \int_\Omega f c_h d\Omega + \frac{1}{2}\int_\Omega \vec{u}_h.\vec{\nabla}(c_h^2) d\Omega. \tag{2.3.4}$$

This is almost the same result as we had in the previous section, and for the same reason we will therefore not get the L^2-stability.

2.4 Simple L^2-stable formulation

Here we mix the two previous formulations, taking half of the first and half of the second formulation. The problem becomes to find $c_h \in H^1(0,T;V_h)$ such that for all $v \in V_h$

$$\int_\Omega \frac{\partial c_h}{\partial t} v d\Omega + D\int_\Omega \vec{\nabla} c_h \vec{\nabla} v d\Omega + \frac{1}{2}\int_\Omega \vec{u}_h.\vec{\nabla} c_h v d\Omega - \frac{1}{2}\int_\Omega \vec{u}_h.\vec{\nabla} v c_h d\Omega = \int_\Omega f v d\Omega. \tag{2.4.1}$$

We check our three conditions:

Mass conservation. We take $v \equiv 1$ and equation (2.4.1) becomes

$$\frac{d}{dt}\int_\Omega c_h d\Omega + \frac{1}{2}\int_\Omega \vec{u}_h.\vec{\nabla} c_h d\Omega = \int_\Omega f d\Omega. \tag{2.4.2}$$

As for the direct formulation we are left with $\int_\Omega \vec{u}_h.\vec{\nabla} c_h d\Omega$, which can not be guaranteed to be equal to zero and therefore spoils the mass conservation.

Conservation of trivial solution. Setting $f = 0$ and $c^0 = 1$ we get

$$\int_\Omega \frac{\partial c_h}{\partial t} d\Omega - \frac{1}{2}\int_\Omega \vec{u}_h.v d\Omega = 0. \tag{2.4.3}$$

Here we have the same problem as for the previous formulation and thus the trivial solution is generally not conserved.

L^2-stability. We set $v = c_h$ in equation (2.4.1) and we obtain

$$\frac{1}{2}\frac{d}{dt}\int_\Omega c_h^2 d\Omega + D\int_\Omega |\vec{\nabla} c_h|^2 d\Omega = \int_\Omega f c_h d\Omega. \tag{2.4.4}$$

Writing this equation using norms and applying the Cauchy-Schwarz inequality we get the desired result,

$$\frac{d}{dt}\|c_h(t)\|_{L^2(\Omega)} \leq \|f(t)\|_{L^2(\Omega)}, \forall t > 0, \tag{2.4.5}$$

because the positive term $\int_\Omega |\vec{\nabla} c_h|^2 d\Omega$ can be dropped. Thus, the result justifies the title.

2.5 Velocity projection on sine and cosine functions

Here we want to project the velocity field $\vec{u}_h$ on divergence free functions formed by a combination of sine and cosine functions, to obtain a projected velocity field $\Pi_{\sin}\vec{u}_h$. We then use the direct weak formulation of Section 2.2 with the modified velocity field.
In order to do this we have to suppose that Ω is a cuboid such that $\vec{x} \in \Omega = [0, L_1] \times [0, L_2] \times [0, L_3]$. We suppose that a projection of $\vec{u}_h$ can be written as

$$\Pi \vec{u}_h(\vec{x}) = \sum_{\vec{k}} u_{\vec{k}} \vec{F}_{\vec{k}}(\vec{x}), \tag{2.5.1}$$

with

$$\vec{F}_{\vec{k}}(\vec{x}) = \begin{pmatrix} \alpha_{\vec{k}} \sin(k_1 \frac{\pi}{L_1} x_1) \cos(k_2 \frac{\pi}{L_2} x_2) \cos(k_3 \frac{\pi}{L_3} x_3) \\ \beta_{\vec{k}} \cos(k_1 \frac{\pi}{L_1} x_1) \sin(k_2 \frac{\pi}{L_2} x_2) \cos(k_3 \frac{\pi}{L_3} x_3) \\ \gamma_{\vec{k}} \cos(k_1 \frac{\pi}{L_1} x_1) \cos(k_2 \frac{\pi}{L_2} x_2) \sin(k_3 \frac{\pi}{L_3} x_3) \end{pmatrix} \tag{2.5.2}$$

and $\vec{x} = (x_1, x_2, x_3) \in [0, L_1] \times [0, L_2] \times [0, L_3], \vec{k} = (k_1, k_2, k_3), k_i = 0, 1, \ldots$. Since we want the velocity field to be divergence free we impose $\operatorname{div} \vec{F}_{\vec{k}}(\vec{x}) = 0$, for every $\vec{x}$, which leads to the condition

$$\frac{k_1}{L_1}\alpha_{\vec{k}} + \frac{k_2}{L_2}\beta_{\vec{k}} + \frac{k_3}{L_3}\gamma_{\vec{k}} = 0. \tag{2.5.3}$$

We also want to have orthonormality of the basis functions, i.e., given $\vec{k}$ and $\vec{j}$ we want to have

$$\int_0^{L_1}\int_0^{L_2}\int_0^{L_3} \vec{F}_{\vec{k}}(\vec{x})\vec{F}_{\vec{j}}(\vec{x})d\Omega = \delta_{k_1,j_1}\delta_{k_2,j_2}\delta_{k_3,j_3}, \tag{2.5.4}$$

where δ_{ij} denotes the Kronecker symbol. This gives a second condition on the coefficients,

$$\alpha_{\vec{k}}^2 + \beta_{\vec{k}}^2 + \gamma_{\vec{k}}^2 = \frac{8}{L_1L_2L_3}. \tag{2.5.5}$$

For each vector $\vec{k}$ verifying $k_1k_2k_3 \neq 0$, two linearly independent nonzero vectors $(\alpha_{\vec{k}}, \beta_{\vec{k}}, \gamma_{\vec{k}})$ which verify the two preceding conditions exist, and thus two basis vectors $\vec{F}_{\vec{k},1}$ and $\vec{F}_{\vec{k},2}$ exist. The coefficients $u_{\vec{k},1}$ and $u_{\vec{k},2}$, which are the components of the projection of $\vec{u}_h$ on the vectors $\vec{F}_{\vec{k},1}$ and $\vec{F}_{\vec{k},2}$ are obtained by computing

$$u_{\vec{k},i} = \int_0^{L_1}\int_0^{L_2}\int_0^{L_3} \vec{u}_h.\vec{F}_{\vec{k},i}d\Omega, \tag{2.5.6}$$

$i = 1, 2$. The frequencies k_i of the sine and cosine functions are chosen such that at least 5 vertices of the mesh per period are available.
We call

$$\Pi_{\sin}\vec{u}_h(\vec{x}) = \sum_{\vec{k}} u_{\vec{k},1}\vec{F}_{\vec{k},1}(\vec{x}) + \sum_{\vec{k}} u_{\vec{k},2}\vec{F}_{\vec{k},2}(\vec{x}) \tag{2.5.7}$$

the projected velocity field.
Replacing $\vec{u}_h$ by $\Pi_{\sin}\vec{u}_h$ in equation (2.2.1) we check if our three conditions are verified:

Mass conservation. Taking $v \equiv 1$ we get

$$\frac{d}{dt}\int_\Omega c_h d\Omega + \int_\Omega \Pi_{\sin}\vec{u}_h.\vec{\nabla}c_h d\Omega = \int_\Omega f d\Omega. \tag{2.5.8}$$

Here by construction we have $\int_\Omega \Pi_{\sin}\vec{u}_h.\vec{\nabla}c_h d\Omega = 0$, hence we ensure mass conservation.

Conservation of trivial solution. The reasoning of Section 2.2 is independent of the velocity field. Therefore the trivial solution is conserved.

L^2-stability. We take $v = c_h$ and we get

$$\frac{1}{2}\frac{d}{dt}\int_\Omega c_h^2 d\Omega + D\int_\Omega |\vec{\nabla}c_h|^2 d\Omega = \int_\Omega f c_h d\Omega - \frac{1}{2}\int_\Omega \Pi_{\sin}\vec{u}_h.\vec{\nabla}(c_h^2)d\Omega. \tag{2.5.9}$$

Again the integral $\int_\Omega \Pi_{\sin}\vec{u}_h.\vec{\nabla}(c_h^2)d\Omega$ vanishes by construction of $\Pi_{\sin}\vec{u}_h$, and then by the argument given in Section 2.4 we obtain the L^2-stability.

Remark 2.5.1. Although this projection on sine and cosine functions has the desired properties, some remarks have to be made.

1. This approach might need a simplification of the geometry of Ω, since the domain has to be a cuboid. In particular, in the alumina problem which we want to solve later on this will not be the case.
2. The projection of $\vec{u}_h$ on the sine and cosine functions modifies the velocity field. In particular, we cannot take any frequencies in the sine and cosine functions because the mesh will not represent high frequencies satisfactorily. We will choose the highest frequency in such a way that there are at least five nodes per period. Hence we are not solving the original problem, and depending on the velocity field the difference could be quite sensible.

2.6 Velocity projection on a divergence free, non-conforming finite element basis

In this section we project the velocity field $\vec{u}_h$ on a divergence free, non-conforming finite element basis to obtain a projected velocity field $\Pi_{\mathrm{FE}}\vec{u}_h$. As in the previous section we will then use the direct weak formulation of Section 2.2 with the modified velocity field.

A three-dimensional, divergence free, non-conforming finite element space W_h has been proposed by Hecht [23]. It is defined by

$$\begin{aligned} W_h = \{\vec{u}_h \in (L^2(\Omega))^3 : \vec{u}_h|_{\mathring{K}} \in (P_1(K))^3, \operatorname{div} \vec{u}_h|_{\mathring{K}} = 0, \forall K \in \mathcal{T}_h; \\ \text{for all faces } T_j \in \mathcal{T}_h : \text{if } K, K' \in \mathcal{T}_h \text{ such that } K \cap K' = T_j \text{ then } \vec{u}_h(\vec{x}^j)|_K = \vec{u}_h(\vec{x}^j)|_{K'}; \\ \int_T \vec{u}_h.\vec{n} d\sigma = 0, \forall T \in \mathcal{G}_h\}, \end{aligned} \quad (2.6.1)$$

where $\vec{x}^j$ is the barycentre of triangle T_j, $\mathcal{T}_h$ is the tetrahedral mesh of Ω and $\mathcal{G}_h$ is the triangular mesh of $\partial\Omega$. The basis functions are associated to the barycentres of the faces and to a subset of the edges. The resulting field

$$\Pi_{\mathrm{FE}}\vec{u}_h = \text{projection of } \vec{u}_h \text{ on } W_h \quad (2.6.2)$$

is piecewise polynomial of degree 1 per element, discontinuous at each face except at the barycentre, and it verifies $\operatorname{div}(\Pi_{\mathrm{FE}}\vec{u}_h) = 0$ per element. For more details on the basis functions, see Hecht [23].

We replace $\vec{u}_h$ by $\Pi_{\mathrm{FE}}\vec{u}_h$ in equation (2.2.1) and we check if our three conditions are verified:

Mass conservation. We set $v \equiv 1$ and we obtain

$$\frac{d}{dt}\int_\Omega c_h d\Omega + \int_\Omega \Pi_{\mathrm{FE}}\vec{u}_h.\vec{\nabla} c_h d\Omega = \int_\Omega f d\Omega. \quad (2.6.3)$$

We compute

$$\int_\Omega \Pi_{\mathrm{FE}}\vec{u}_h.\vec{\nabla} c_h d\Omega = \sum_{K\in\mathcal{T}_h}\int_K c_h \operatorname{div}(\Pi_{\mathrm{FE}}\vec{u}_h) d\Omega + \sum_{K\in\mathcal{T}_h}\int_{\partial K} c_h \Pi_{\mathrm{FE}}\vec{u}_h.\vec{n} d\sigma, \quad (2.6.4)$$

where we have to separate the integral over Ω in the sum of integrals over the tetrahedrons K because the projected velocity $\Pi_{\mathrm{FE}}\vec{u}_h$ is only in $(L^2(\Omega))^3$. The first term of the right hand side equals zero because $\operatorname{div}(\Pi_{\mathrm{FE}}\vec{u}_h) = 0$ in each tetrahedron. But the second term of the right hand side is not equal to zero, because the velocity is continuous only at the barycentres of the triangles which form the boundary of the tetrahedrons, but not on the whole triangles. Therefore, we find that

$$\int_\Omega \Pi_{\mathrm{FE}}\vec{u}_h.\vec{\nabla} c_h d\Omega \neq 0 \quad (2.6.5)$$

in general, implying that the total mass will not be conserved.

Conservation of trivial solution. The reasoning of Section 2.2 is independent of the velocity field. Therefore the trivial solution is conserved.

L^2-stability. Taking $v = c_h$ we have

$$\frac{1}{2}\frac{d}{dt}\int_\Omega c_h^2 d\Omega + D\int_\Omega |\vec{\nabla} c_h|^2 d\Omega = \int_\Omega f c_h d\Omega - \frac{1}{2}\int_\Omega \Pi_{\mathrm{FE}}\vec{u}_h.\vec{\nabla}(c_h^2) d\Omega. \tag{2.6.6}$$

Using the same argument as for the mass conservation the integral $\int_\Omega \Pi_{\mathrm{FE}}\vec{u}_h.\vec{\nabla}(c_h^2)d\Omega$ does in general not vanish and thus we do not get the L^2-stability.

Remark 2.6.1. In addition to the fact that the projected velocity field $\Pi_{\mathrm{FE}}\vec{u}_h$ is a priori not better than the original velocity $\vec{u}_h$ it has to be stressed that the number of basis functions of W_h is much larger than the number of basis functions of V_h. Hence, the projection of the velocity can become difficult or impossible on a fine mesh. We also note that the velocity field is modified by the projection, and thus the problem we solve is not equivalent to the original problem.

2.7 An approximated Helmholtz decomposition of the velocity field

There is a decomposition theorem which states that every vector field $\vec{u} \in (L^2(U))^3$, for some open domain $U \in \mathbb{R}^3$, can be written as the sum of a divergence free part and an irrotational part,

$$\vec{u} = -\vec{\nabla} q + \operatorname{curl}\vec{\phi} \tag{2.7.1}$$

where $q \in H^1(U)/\mathbb{R}$ is the only solution of

$$\int_U \vec{\nabla} q.\vec{\nabla} v dU = \int_U \vec{u}.\vec{\nabla} v dU, \quad \forall v \in H^1(U), \tag{2.7.2}$$

and $\vec{\phi} \in (H^1(U))^3$ satisfies $\mathrm{div}(\vec{\phi}) = 0$ [17]. This decomposition is also known as the Helmholtz decomposition. Here we want to use this property to decompose our velocity field $\vec{u}_h$ and replace it by the divergence free part.
We define

$$\vec{w}_h = \vec{u}_h - \vec{\nabla}\psi_h, \tag{2.7.3}$$

where $\psi_h \in V_h$ verifies:

$$\int_\Omega \vec{\nabla}\psi_h.\vec{\nabla} v d\Omega = \int_\Omega \vec{u}_h.\vec{\nabla} v d\Omega, \quad \forall v \in V_h. \tag{2.7.4}$$

Here $\vec{u}_h = \vec{w}_h + \vec{\nabla}\psi_h$ can be interpreted as an approximated Helmholtz decomposition of $\vec{u}_h$. If $\vec{u}_h$ satisfies (2.1.4) then $\vec{\nabla}\psi_h = 0$ and $\vec{u}_h = \vec{w}_h$.
Again we replace $\vec{u}_h$ by $\vec{w}_h$ in equation (2.2.1) and we check our three conditions:

Mass conservation. Taking $v \equiv 1$ we get

$$\frac{d}{dt}\int_\Omega c_h d\Omega + \int_\Omega w_h.\vec{\nabla} c_h d\Omega = \int_\Omega f d\Omega. \tag{2.7.5}$$

We have $w_h = u_h - \vec{\nabla}\psi_h$ and using equation (2.7.4) we see that $\int_\Omega w_h.\vec{\nabla} c_h d\Omega = 0$, which implies the mass conservation.

Conservation of trivial solution. The reasoning of Section 2.2 is independent of the velocity field. Therefore the trivial solution is conserved.

L^2-stability. We set $v = c_h$. We obtain

$$\frac{1}{2}\frac{d}{dt}\int_\Omega c_h^2 d\Omega + D\int_\Omega |\vec{\nabla} c_h|^2 d\Omega = \int_\Omega f c_h d\Omega - \frac{1}{2}\int_\Omega w_h.\vec{\nabla}(c_h^2)d\Omega. \tag{2.7.6}$$

Here the integral $\int_\Omega w_h.\vec{\nabla}(c_h^2)d\Omega$ does not vanish, because when we compute $\vec{\nabla}\psi_h$ we take the test function $v \in V_h$, which is only in P_1. Thus with the present discretization we do not have L^2-stability, but if we take test functions which are at least in P_2 we can get it.

Remark 2.7.1. Observe that if we discretize (2.7.4) with ψ_h in V_h, $\vec{\nabla}\psi_h$ is constant in each tetrahedron and discontinuous on the boundaries. This implies that the velocity $\vec{w}_h$ is discontinuous. We also note that we change the velocity field when we pass from $\vec{u}_h$ to $\vec{w}_h$ and are thus not solving the original problem.

2.8 Meanfree mass conserving formulation

The following scheme is the origin of the scheme proposed in [25]. We consider here the following modified problem: Find $c_h \in H^1(0,T;V_h)$ such that for all $v \in V_h$

$$\int_\Omega \frac{\partial c_h}{\partial t} v d\Omega + \int_\Omega \vec{u}_h.\vec{\nabla} c_h (v - \bar{v}) d\Omega + D\int_\Omega \vec{\nabla} c_h.\vec{\nabla} v d\Omega = \int_\Omega f v d\Omega, \tag{2.8.1}$$

with $\bar{v} = \frac{1}{|\Omega|}\int_\Omega v d\Omega$ the mean of v over Ω. Remark that if the approximated velocity field $\vec{u}_h$ verifies (2.1.4), then $\int_\Omega \vec{u}_h.\vec{\nabla} c_h \bar{v} d\Omega = 0$ and problem (2.8.1) is equivalent to problem (2.2.1). We check our three conditions:

Mass conservation. We take $v \equiv 1$ in equation (2.8.1), getting

$$\frac{d}{dt}\int_\Omega c_h d\Omega = \int_\Omega f d\Omega, \tag{2.8.2}$$

which gives the mass conservation. The velocity term cancels out because here $v = 1 = \bar{v}$.

Conservation of trivial solution. If we suppose that $f = 0$ and $c^0 = 1$ then $c_h(\vec{x}, t) = c^0(\vec{x})$ is the unique solution of (2.8.1).

L^2-stability. Let $v = c_h$ in equation (2.8.1). We obtain

$$\frac{1}{2}\frac{d}{dt}\int_\Omega c_h^2 d\Omega + D\int_\Omega |\vec{\nabla} c_h|^2 d\Omega = \int_\Omega f c_h d\Omega - \frac{1}{2}\int_\Omega \vec{u}_h.\vec{\nabla}(c_h^2) d\Omega + \bar{c}_h \int_\Omega \vec{u}_h.\vec{\nabla} c_h d\Omega. \tag{2.8.3}$$

Since the last two integrals do not cancel we can not assert the L^2-stability.

This seems to be an elegant way to get the mass conservation and the conservation of the trivial solution. But the weak formulation contains the mean of the test function, which implies that the linear system arising of the discretization will contain a full matrix. For a finite element discretization this is certainly not desirable. In what follows, we will give two algorithms of how we can find a solution to this weak formulation without having to cope with the full matrix.
For these algorithms we first discretize equation (2.8.1) in time with the implicit Euler scheme. The problem becomes to find $c_h^{n+1} \in V_h$ such that for all $v \in V_h$

$$\int_\Omega \frac{c_h^{n+1} - c_h^n}{\Delta t} v d\Omega + \int_\Omega \vec{u}_h.\vec{\nabla} c_h^{n+1} (v - \bar{v}) d\Omega + D\int_\Omega \vec{\nabla} c_h^{n+1}.\vec{\nabla} v d\Omega = \int_\Omega f(t^{n+1}) v d\Omega. \tag{2.8.4}$$

Direct algorithm using GMRES

If we directly discretize the weak formulation (2.8.4) then we have to deal with the term

$$\int_\Omega \vec{u}_h.\vec{\nabla} c_h^{n+1} \bar{v} d\Omega. \tag{2.8.5}$$

$\bar{v}$ is a constant defined on the whole mesh and it is not equal to zero when v is a finite element function. Discretizing this convection term in space results in a full matrix. It is not reasonable to store this matrix, but this is not necessary. We want to solve some system

$$A\vec{x} = \vec{b}, \tag{2.8.6}$$

with A the sparse matrix resulting from the discretization of equation (2.8.4) without the term of equation (2.8.5). If we subtract the term of equation (2.8.5) we can write the discrete problem as

$$A\vec{x} + B\vec{x} = \vec{d}, \tag{2.8.7}$$

where B is the full matrix resulting from the discretization of the additional term containing $\bar{v}$. If we solve this linear system using GMRES algorithm ([33]), all we have to do is to compute matrix vector products. Let $\vec{x}$ be a vector. Computing $A\vec{x}$ is easy since A is supposed to be a sparse matrix. Let's discretize the term (2.8.5). To do this, let $v_1, v_2, \ldots, v_N$ be the finite element basis of V_h. Writing $c_h^{n+1} = \sum_{j=1}^N c_j v_j$, $g_j = \int_\Omega \vec{u}_h.\vec{\nabla} v_j d\Omega$, $\bar{v}_j = \frac{1}{|\Omega|}\int_\Omega v_j d\Omega$ we have:

$$\int_\Omega \vec{u}_h.\vec{\nabla} c_h^{n+1} \bar{v}_i d\Omega = \int_\Omega \sum_{j=1}^N c_j \vec{u}_h.\vec{\nabla} v_j \bar{v}_i d\Omega = \bar{v}_i \sum_{j=1}^N c_j \int_\Omega \vec{u}_h.\vec{\nabla} v_j d\Omega = \bar{v}_i \sum_{j=1}^N c_j g_j. \tag{2.8.8}$$

Defining $\vec{v}_m = (\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_N)$, $\vec{g} = (g_1, g_2, \ldots, g_N)$ and $\vec{c} = (c_1, c_2, \ldots, c_N)$ we can write the discretization of (2.8.5) as

$$(\vec{v}_m.\vec{g}^T)\vec{c}. \tag{2.8.9}$$

Note that $\vec{v}_m.\vec{g}^T$ is a $N \times N$ matrix, in fact the matrix B, with elements $B_{ij} = \bar{v}_i g_j$. The product $B\vec{x}$ can be written as

$$(B\vec{x})_i = \sum_{j=1}^N B_{ij} x_j = \sum_{j=1}^N \bar{v}_i g_j x_j = \bar{v}_i \sum_{j=1}^N g_j x_j, \quad i = 1, \ldots, N. \tag{2.8.10}$$

Hence, in order to get $B\vec{x}$ we can first compute $a = \vec{g}.\vec{x}$ and then $\vec{v}_m a = B\vec{x}$, and we only need to store two vectors of length N to be able to do this. The drawback of this method is that it does not work if we want to solve the linear system with a direct method, or some other method which explicitly needs the complete matrix.

Generally applicable algorithm

A more generally applicable algorithm, where the linear system can even be solved with methods which explicitly need the matrix, is the following:

Algorithm 2.8.1. Let $\tilde{c}_h^n = c_h^n - \bar{c}_h^n$ and $\tilde{f}^{n+1} = f(t^{n+1}) - \bar{f}(t^{n+1})$. We find $\hat{c}^{n+1} \in V_h$ verifying for each $v \in V_h$

$$\int_\Omega \frac{\hat{c}^{n+1} - \tilde{c}_h^n}{\Delta t} v d\Omega + \int_\Omega \vec{u}_h.\vec{\nabla} \hat{c}^{n+1} v d\Omega + D \int_\Omega \vec{\nabla} \hat{c}^{n+1}.\vec{\nabla} v d\Omega = \int_\Omega \tilde{f}^{n+1} v d\Omega. \tag{2.8.11}$$

Then we set

$$c_h^{n+1} = \hat{c}^{n+1} - \bar{\hat{c}}^{n+1} + \bar{c}_h^n + \Delta t \bar{f}(t^{n+1}). \tag{2.8.12}$$

Here the sparsity pattern of the matrix is still the same as for the discretization of problem (2.2.1), i.e., only neighbouring nodes create entries in the matrix.
We show now that the solution of Algorithm 2.8.1 is equivalent to the solution of problem (2.8.4). From (2.8.12) we get

$$\hat{c}^{n+1} = c_h^{n+1} + \bar{\hat{c}}^{n+1} - \bar{c}_h^n - \Delta t \bar{f}(t^{n+1}). \tag{2.8.13}$$

Replacing $\hat{c}^{n+1}$ in (2.8.11) we get

$$\begin{aligned}\int_\Omega \frac{c_h^{n+1} + \bar{\hat{c}}^{n+1} - \bar{c}_h^n - \tilde{c}_h^n - \Delta t \bar{f}(t^{n+1})}{\Delta t} v d\Omega + \int_\Omega \vec{u}_h . \vec{\nabla} \hat{c}^{n+1} v d\Omega + D \int_\Omega \vec{\nabla} \hat{c}^{n+1} \vec{\nabla} v d\Omega \\ = \int_\Omega f(t^{n+1}) v d\Omega - \int_\Omega \bar{f}(t^{n+1}) v d\Omega.\end{aligned} \tag{2.8.14}$$

Noting that $\vec{\nabla} \hat{c}^{n+1} = \vec{\nabla} c_h^{n+1}$ since the other terms in (2.8.12) are constants, substituting $\bar{c}_h^n + \tilde{c}_h^n$ by c_h^n and adding $\int_\Omega \bar{f}(t^{n+1}) v d\Omega$ we are left with

$$\int_\Omega \frac{c_h^{n+1} + \bar{\hat{c}}^{n+1} - c_h^n}{\Delta t} v d\Omega + \int_\Omega \vec{u}_h . \vec{\nabla} c_h^{n+1} v d\Omega + D \int_\Omega \vec{\nabla} c_h^{n+1} \vec{\nabla} v d\Omega = \int_\Omega f(t^{n+1}) v d\Omega. \tag{2.8.15}$$

Now we set $v \equiv 1$ in (2.8.11). We obtain

$$\frac{|\Omega|}{\Delta t} \bar{\hat{c}}^{n+1} = - \int_\Omega \vec{u}_h . \vec{\nabla} c_h^{n+1} d\Omega, \tag{2.8.16}$$

i.e.,

$$\int_\Omega \frac{\bar{\hat{c}}^{n+1}}{\Delta t} v d\Omega = \frac{\bar{\hat{c}}^{n+1}}{\Delta t} \int_\Omega v d\Omega = \frac{|\Omega|}{\Delta t} \bar{\hat{c}}^{n+1} \bar{v} = - \int_\Omega \vec{u}_h . \vec{\nabla} c_h^{n+1} \bar{v} d\Omega. \tag{2.8.17}$$

Substituting this in (2.8.15) and rearranging the terms we get (2.8.4). Thus, the solution of Algorithm 2.8.1 is the same as the solution of (2.8.4).

Remark 2.8.2. We want to solve a convection-diffusion problem with a velocity field which does not verify conditions (2.1.4). The formulation presented in this section shows some nice properties, even if it does not produce a solution which is L^2-stable. However, looking at Algorithm 2.8.1 we see that the mass conservation is attained simply by redistributing the mass error over the whole domain, i.e., by correcting the local errors globally. It is clear that if the approximated velocity field almost satisfies conditions (2.1.4), then already the local errors will be small and the global correction applied by Algorithm 2.8.1 will make them almost invisible. But the initial "error" of the velocity field will not disappear, and especially if conditions (2.1.4) are strongly violated the solution could show a strange behaviour at large times. The formulation presented here does not wipe out the initial error of $\vec{u}_h$, it only attains that the error will not be visible in the mass balance.

2.9 Meanfree L^2-stable mass conserving formulation

Finally we consider the following problem, which was proposed in [25]: Find $c_h \in H^1(0,T;V_h)$ such that for all $v \in V_h$

$$\int_\Omega \frac{\partial c_h}{\partial t} v d\Omega + D \int_\Omega \vec{\nabla} c_h \vec{\nabla} v d\Omega + \frac{1}{2} \int_\Omega \vec{u}_h . \vec{\nabla} c_h (v - \bar{v}) d\Omega - \frac{1}{2} \int_\Omega \vec{u}_h . \vec{\nabla} v (c_h - \bar{c}_h) d\Omega = \int_\Omega f v d\Omega. \tag{2.9.1}$$

Again we denote by $\bar{v} = \frac{1}{|\Omega|} \int_\Omega v d\Omega$ the mean of a function v over Ω. We note that if the approximated velocity field $\vec{u}_h$ verifies the conditions (2.1.4) then the formulation (2.9.1) is

equivalent to the direct formulation (2.2.1) since the integrals containing $\bar{v}$ and $\bar{c}_h$ will vanish, and the original convection term can be obtained by integration by parts.
We check the three conditions:

Mass conservation. Let $v \equiv 1$ in equation (2.9.1). We obtain

$$\int_\Omega \frac{\partial c_h}{\partial t} v d\Omega = \int_\Omega f v d\Omega. \tag{2.9.2}$$

Thus, mass conservation is attained.

Conservation of trivial solution. If we suppose that $f = 0$ and $c^0 = 1$ then $c_h(\vec{x}, t) = c^0(\vec{x})$ is the unique solution of (2.9.1).

L^2-stability. We take $v = c_h$ in (2.9.1) and we get

$$\frac{1}{2}\frac{d}{dt}\int_\Omega c_h^2 d\Omega + D\int_\Omega |\vec{\nabla} c_h|^2 d\Omega = \int_\Omega f c_h d\Omega. \tag{2.9.3}$$

Writing this equation using norms and applying Cauchy-Schwarz's inequality we get the desired result,

$$\frac{d}{dt}\|c_h(t)\|_{L^2(\Omega)} \leq \|f(t)\|_{L^2(\Omega)}, \forall t > 0, \tag{2.9.4}$$

because the positive term $\int_\Omega |\vec{\nabla} c_h|^2 d\Omega$ can be dropped. Hence the solution is L^2-stable as we state in the title of the section.

So here we have a formulation to which the solution is L^2-stable and mass conserving and which admits the trivial solution. As in the last section the formulation contains mean functions which are nonzero over the whole domain, implying that straightforward discretization of the weak formulation leads to a full matrix. Again we will give two different algorithms to circumvent the problem, and again one of them will be useful only with special solvers of linear systems while the other leads to a linear system which can be solved by a broader class of methods. We discretize first the problem (2.9.14) in time by the implicit Euler method. The problem becomes to find $c_h^{n+1} \in V_h$ such that for all $v \in V_h$

$$\int_\Omega \frac{c_h^{n+1} - c_h^n}{\Delta t} v d\Omega + D\int_\Omega \vec{\nabla} c_h^{n+1} \vec{\nabla} v d\Omega + \frac{1}{2}\int_\Omega \vec{u}_h . \vec{\nabla} c_h^{n+1}(v - \bar{v}) d\Omega$$
$$- \frac{1}{2}\int_\Omega \vec{u}_h . \vec{\nabla} v (c_h^{n+1} - \bar{c}_h^{n+1}) d\Omega = \int_\Omega f(t^{n+1}) v d\Omega. \tag{2.9.5}$$

Before continuing by the algorithms of how to solve this problem, we give some properties of the solution of this problem. For a functional space W, let $\widetilde{W} = \{g \in W : \int_\Omega g dx = 0\}$. Let $R_h : \widetilde{H}^1(\Omega) \to \widetilde{V}_h$ be the elliptic operator defined by

$$\int_\Omega \vec{\nabla}(\eta - R_h \eta) . \vec{\nabla} \psi d\Omega = 0, \quad \forall \psi \in \widetilde{V}_h, \forall \eta \in \widetilde{H}^1(\Omega). \tag{2.9.6}$$

In [25] we prove the following

Theorem 2.9.1. *If $c \in C^1([0,T]; H^2(\Omega)) \cap C^2([0,T]; L^2(\Omega))$ and if there exists a constant C independent of h and n such that $\|c_h^n\|_{L^\infty(\Omega)} \leq C$; if in addition there exists another constant C_2 such that $\|\vec{u} - \vec{u}_h\|_{L^2(\Omega)} + h\|\vec{\nabla}(\vec{u} - \vec{u}_h)\|_{L^2(\Omega)} \leq Ch^2$, $c^0 \in H^2(\Omega)$ and $c_h^0 = \bar{c}^0 + R_h(c^0 - \bar{c}^0)$. Then there exists a constant C_2 independent of h which satisfies*

$$\|c(t^n) - c_h^n\|_{L^2(\Omega)} + \|\vec{\nabla}(c(t^n) - c_h^n)\|_{L^2(\Omega)} \leq C_2(h + \Delta t), \quad \forall 1 \leq n \leq N. \tag{2.9.7}$$

Here, $N = T/\Delta t$.

Hence, the convergence of the numerical solution is ensured.
Now we return to the algorithms that solve Problem (2.9.5).

Direct algorithm using GMRES

We consider the following algorithm:

Algorithm 2.9.2. Let $\tilde{c}_h^n = c_h^n - \bar{c}_h^n$ and $\tilde{f}^{n+1} = f(t^{n+1}) - \bar{f}(t^{n+1})$. We look for $\hat{c}^{n+1} \in V_h$ such that for all $v \in V_h$

$$\int_\Omega \frac{\hat{c}^{n+1} - \tilde{c}_h^n}{\Delta t} v d\Omega + D \int_\Omega \vec{\nabla} \hat{c}^{n+1} \vec{\nabla} v d\Omega + \frac{1}{2} \int_\Omega \vec{u}_h . \vec{\nabla} \hat{c}^{n+1} (v - \bar{v}) d\Omega - \frac{1}{2} \int_\Omega \vec{u}_h . \vec{\nabla} v \hat{c}^{n+1} d\Omega = \int_\Omega \tilde{f}^{n+1} v d\Omega. \quad (2.9.8)$$

We then set

$$c_h^{n+1} = \hat{c}^{n+1} + \bar{c}^n + \Delta t \bar{f}(t^{n+1}). \quad (2.9.9)$$

To solve equation (2.9.8) we use the same trick as in Section 2.8. Therefore one has to use GMRES or a similar solver to solve the linear system.
It remains to show that the solution of Algorithm 2.9.2 is equivalent to the solution of problem (2.9.5). To this end we write equation (2.9.9) as

$$\hat{c}^{n+1} = c_h^{n+1} - \bar{c}^n - \Delta t \bar{f}(t^{n+1}). \quad (2.9.10)$$

We substitute $\hat{c}^{n+1}$ in (2.9.8), getting, after some simplifications similar to those done in the previous section,

$$\int_\Omega \frac{c_h^{n+1} - c_h^n}{\Delta t} v d\Omega + D \int_\Omega \vec{\nabla} c_h^{n+1} \vec{\nabla} v d\Omega + \frac{1}{2} \int_\Omega \vec{u}_h . \vec{\nabla} c_h^{n+1} (v - \bar{v}) d\Omega - \frac{1}{2} \int_\Omega \vec{u}_h . \vec{\nabla} v (c_h^{n+1} - \bar{c}^n - \Delta t \bar{f}(t^{n+1})) d\Omega = \int_\Omega f(t^{n+1}) v d\Omega. \quad (2.9.11)$$

Starting from (2.9.9) we compute

$$\bar{c}_h^{n+1} = \bar{\hat{c}}^{n+1} + \bar{c}^n + \Delta t \bar{f}(t^{n+1}) = \bar{c}^n + \Delta t \bar{f}(t^{n+1}), \quad (2.9.12)$$

where the last equality is true because by setting $v \equiv 1$ in (2.9.8) we get

$$\bar{\hat{c}}^{n+1} = 0. \quad (2.9.13)$$

Replacing $\bar{c}^n + \Delta t \bar{f}(t^{n+1})$ by $\bar{c}_h^{n+1}$ in (2.9.11) we get equation (2.9.5), proving that the Algorithm 2.9.2 leads to the same solution as problem (2.9.5).

Generally applicable algorithm

In case we want to use for example a direct linear solver to solve the discretized problem (2.9.1) we propose the following algorithm (where a Lagrange multiplier is used to guarantee that $\bar{v} = \frac{1}{|\Omega|} \int_\Omega v d\Omega$, see [25]).

Algorithm 2.9.3. Let $\tilde{c}_h^n = c_h^n - \bar{c}_h^n$ and $\tilde{f}^{n+1} = f(t^{n+1}) - \bar{f}(t^{n+1})$. We look for $\hat{c}^{n+1} \in V_h$ and $\alpha \in \mathbb{R}$ such that for all $v \in V_h$

$$\int_\Omega \frac{\hat{c}^{n+1} - \tilde{c}_h^n}{\Delta t} v d\Omega + D \int_\Omega \vec{\nabla} \hat{c}^{n+1} \vec{\nabla} v d\Omega + \frac{1}{2} \int_\Omega \vec{u}_h . \vec{\nabla} \hat{c}^{n+1} v d\Omega - \frac{1}{2} \int_\Omega \vec{u}_h . \vec{\nabla} v \hat{c}^{n+1} d\Omega + \alpha \int_\Omega v d\Omega = \int_\Omega \tilde{f}^{n+1} v d\Omega, \quad (2.9.14)$$

and

$$\int_\Omega \hat{c}^{n+1} d\Omega = 0. \tag{2.9.15}$$

Then we set

$$c_h^{n+1} = \hat{c}^{n+1} + \bar{c}_h^n + \Delta t \bar{f}(t^{n+1}). \tag{2.9.16}$$

If c_h is a vector of length N, then the discretization of problem (2.9.14)-(2.9.15) leads to a $(N+1) \times (N+1)$-matrix with an additional unknown α. Taking $v \equiv 1$ in (2.9.14) we obtain

$$\alpha = -\frac{1}{2|\Omega|} \int_\Omega \vec{u}_h . \vec{\nabla} \hat{c}^{n+1} d\Omega, \tag{2.9.17}$$

hence α will be zero if the velocity field $\vec{u}_h$ verifies conditions (2.1.4).
In order to show the equivalence of the solutions of Algorithm 2.9.3 and of problem (2.9.5) we rearrange (2.9.16) to obtain

$$\hat{c}^{n+1} = c_h^{n+1} - \bar{c}^n - \Delta t \bar{f}(t^{n+1}). \tag{2.9.18}$$

We substitute this expression in (2.9.14), and after some simplifications we get

$$\begin{aligned} \int_\Omega \frac{c_h^{n+1} - c_h^n}{\Delta t} v d\Omega + D \int_\Omega \vec{\nabla} c_h^{n+1} \vec{\nabla} v d\Omega + \frac{1}{2} \int_\Omega \vec{u}_h . \vec{\nabla} c_h^{n+1} v d\Omega \\ - \frac{1}{2} \int_\Omega \vec{u}_h . \vec{\nabla} v (c_h^{n+1} - \bar{c}^n - \Delta t \bar{f}(t^{n+1})) d\Omega + \alpha \int_\Omega v d\Omega = \int_\Omega f(t^{n+1}) v d\Omega. \end{aligned} \tag{2.9.19}$$

We use (2.9.16) and (2.9.15) to compute

$$\bar{c}_h^{n+1} = \bar{c}^n + \Delta t \bar{f}(t^{n+1}), \tag{2.9.20}$$

and (2.9.17) to obtain

$$\alpha \int_\Omega v d\Omega = -\frac{1}{2} \int_\Omega \vec{u}_h . \vec{\nabla} c_h^{n+1} \bar{v} d\Omega, \tag{2.9.21}$$

and substituting these two expressions in (2.9.19) we get to (2.9.5), which is what we wanted.

Remark 2.9.4. Similar considerations as given in the remark at the end of the previous section can be made. The algorithms presented here reach the goals that we defined at the beginning of the chapter. They move the visible defaults of the solution of the convection-diffusion equation away from the mass conservation or the L^2-stability. But these defaults will appear elsewhere, and depending on the properties one wants the solution to fulfil this might be more troublesome.

Remark 2.9.5. A similar algorithm to algorithm 2.9.3 is derived in [25] for the case when the Neumann boundary conditions are replaced by Robin conditions,

$$D \frac{\partial c}{\partial n} = \xi (c_r - c), \text{ on } \partial\Omega \times (0, T), \tag{2.9.22}$$

where ξ is a positive parameter and c_r is a given reference value. The same goals as presented here are attained for this problem using this algorithm.

2.10 Conclusion

In the beginning of this chapter we stated that we wanted a solution to the convection-diffusion Problem 2.1.1 which has the three properties called mass conservation, conservation of the trivial solution and L^2-stability. Of all the formulations that we proposed, only the solution of two of them will reach this aim using linear finite elements, the direct formulation with a projected velocity field on sine and cosine functions, and the last formulation. The problem of the projection of the velocity field is clearly that the domain Ω has to be a cuboid, which is not true for the alumina problem. Therefore this possibility is not really useful. On the other hand the formulation presented in this section does not have any restrictions. We therefore decide to work with this formulation when we solve the alumina Problem 1.2.1. This choice is also justified by the numerical results given in Section 5.2.

Chapter 3

Discretization

In this chapter we will discretize the mathematical problem presented in the first chapter. We will start in the first section by computing approximations of c and n_p in time. In Section 3.2 we will discretize the computation of n_p in the radius variable R. Finally in Section 3.3 we compute finite elements approximations of c and n_p in space on a tetrahedral mesh $\mathcal{T}_h$ of Ω_{el}.

3.1 Discretization in time

Let us choose a discretization of the time t by choosing a time step Δt and setting $t_n = n\Delta t$, for $n = 0, 1, 2, \dots$. If $c^k(\vec{x}), n_p^k(\vec{x}, R)$ are known approximations of $c(\vec{x}, t_k)$ and $n_p(\vec{x}, R, t_k)$ at time t_k, we compute c^{k+1} and n_p^{k+1} at time $t_{k+1} = t_k + \Delta t$ as follows. For each time step Δt we solve successively the equation for the particles and the equation for the concentration. For solving the first equation we use a splitting method in time in order to separate the space-time part from the radius-time part in equation (1.2.20). Numerically we compute:

- Convection equation in space discretized in time on (t_k, t_{k+1}),

$$\frac{\tilde{n}_p^{k+1}(\vec{x}, R) - n_p^k(\vec{x}, R)}{\Delta t} + \vec{u}(\vec{x}).\vec{\nabla}\tilde{n}_p^{k+1}(\vec{x}, R) = S(\vec{x}, R, t_{k+1}), \quad \vec{x} \in \Omega_{\text{el}}, R > 0, \tag{3.1.1}$$

  where $\tilde{n}_p^{k+1}(\vec{x}, R)$ is a prediction of the quantity $n_p^{k+1}(\vec{x}, R)$.

- Particle dissolution (not discretized on (t_k, t_{k+1})),

$$\frac{\partial n_p(\vec{x}, R, t)}{\partial t} + \frac{\partial}{\partial R}\Big(n_p(\vec{x}, R, t) f(R, c^k(\vec{x}))\Big) = 0, \quad \vec{x} \in \Omega_{\text{el}}, R > 0, t \in (t_k, t_{k+1}), \tag{3.1.2}$$

$$n_p(\vec{x}, R, t_k) = \tilde{n}_p^{k+1}(\vec{x}, R). \tag{3.1.3}$$

  We set $n_p^{k+1}(\vec{x}, R) = n_p(\vec{x}, R, t_{k+1})$.

- Source terms of the convection-diffusion equation at time t_k,

$$\dot{q}_1^k(\vec{x}) = -4\pi\frac{\rho}{M}\int_0^\infty n_p^{k+1}(\vec{x}, R) f(R, c^k(\vec{x})) R^2 dR, \tag{3.1.4}$$

$$\dot{q}_2^k = -\frac{I}{6FV}, \tag{3.1.5}$$

$$\dot{Q}^k(\vec{x}) = \dot{q}_1^k(\vec{x}) + \dot{q}_2^k, \tag{3.1.6}$$

- Convection-diffusion equation discretized in time on (t_k, t_{k+1}),

$$\frac{c^{k+1}(\vec{x}) - c^k(\vec{x})}{\Delta t} + \vec{u}(\vec{x}).\vec{\nabla} c^{k+1}(\vec{x}) - D\Delta c^{k+1}(\vec{x}) = \dot{Q}^k(\vec{x}), \quad \text{in } \Omega_{\text{el}}, \tag{3.1.7}$$

$$\vec{\nabla} c^{k+1}(\vec{x}).\vec{n}(\vec{x}) = 0, \qquad \text{on } \partial\Omega_{\text{el}}. \tag{3.1.8}$$

To compute the particle dissolution (3.1.2), we fix $\vec{x} \in \Omega_{\text{el}}$ and to simplify the notation we introduce

$$\phi(R,t) = n_p(\vec{x}, R, t), \tag{3.1.9}$$

$$g(R) = f(R, c^k(\vec{x})), \tag{3.1.10}$$

Equation (3.1.2) becomes

$$\frac{\partial \phi}{\partial t}(R,t) + \frac{\partial}{\partial R}\left(g(R)\phi(R,t)\right) = 0, \tag{3.1.11}$$

which we solve by the method of characteristics [30]. Let

$$\dot{R}(t) = g(R(t)), \quad t \in [t_k, t_{k+1}]. \tag{3.1.12}$$

Setting $\psi(t) = \phi(R(t), t)$ we get, if $\dot{\psi}(t) = \frac{d\psi(t)}{dt}$:

$$\dot{\psi}(t) = \frac{\partial \phi}{\partial t}(R(t),t) + \frac{\partial \phi}{\partial R}(R(t),t)\dot{R}(t) = \frac{\partial \phi}{\partial t}(R(t),t) + \frac{\partial \phi}{\partial R}(R(t),t)g(R(t)), \tag{3.1.13}$$

and using (3.1.11) we obtain

$$\dot{\psi}(t) = -\psi(t)\frac{\partial g}{\partial R}(R(t)). \tag{3.1.14}$$

Solving this equation we find

$$\psi(t_{k+1}) = \psi(t_k) \exp\left(-\int_{t_k}^{t_{k+1}} \frac{\partial g}{\partial R}(R(s))ds\right). \tag{3.1.15}$$

Thus

$$n_p^{k+1}(\vec{x}, R(t_{k+1})) = \tilde{n}_p^{k+1}(\vec{x}, R(t_k)) \exp\left(-\int_{t_k}^{t_{k+1}} \frac{\partial f}{\partial R}(R(s), c^k(\vec{x}))ds\right), \tag{3.1.16}$$

with

$$\dot{R}(t) = f(R(t), c^k(\vec{x})), \quad t \in [t_k, t_{k+1}], \tag{3.1.17}$$

$$R(t_k) \text{ given, for every } \vec{x} \in \Omega_{\text{el}}. \tag{3.1.18}$$

3.2 Discretization in R

For the discretization in the radius R we introduce m radii,

$$0 < R_1 < R_2 < \cdots < R_m. \tag{3.2.1}$$

We set $R_0 = 0$. The particle density n_p^k is approximated by

$$n_{p,j}^k(\vec{x}) \approx n_p^k(\vec{x}, R_j) \tag{3.2.2}$$

The source term of the convection-diffusion equation $\dot{q}_1^k$, defined in equation (3.1.4), is approximated by

$$\dot{q}_1^k(\vec{x}) \approx -4\pi\frac{\rho}{M}\sum_{i=1}^{m} n_{p,i}^{k+1}(\vec{x}) f(R_i, c^k(\vec{x})) R_i^2 \Delta R_i, \tag{3.2.3}$$

where $\Delta R_i = R_i - R_{i-1}$.

To discretize the particle dissolution we take into account equations (3.1.12) and (3.1.15). Suppose that the radius of a particle at position $\vec{x} \in \Omega_{\mathrm{el}}$ and time t_{k+1} is R_j^{k+1}. At time t_k the radius of this particle was, using (3.1.12) with first order accuracy, (backwards method of characteristics of order 1)

$$\tilde{R}_j^k = R_j^{k+1} - \Delta t g(R_j^{k+1}). \tag{3.2.4}$$

Again with first order accuracy we get from (3.1.15)

$$\psi(t_{k+1}) \approx \psi(t_k)\exp\left(-\Delta t\frac{\partial g}{\partial R}(R_j^{k+1})\right) \approx \psi(t_k)\left(1-\Delta t\frac{\partial g}{\partial R}(R_j^{k+1})\right). \tag{3.2.5}$$

We compute $\psi(t_k) = \phi(R(t_k))$ by interpolation. Let s_j be such that $R_{s_j} \le \tilde{R}_j^k < R_{s_j+1}$. Since $\psi(t) = \phi(R(t),t) = n_p(\vec{x}, R(t), t), \forall \vec{x} \in \Omega_{\mathrm{el}}$, we have

$$\psi(t_k) \approx \frac{R_{s_j+1} - \tilde{R}_j^k}{R_{s_j+1} - R_{s_j}}\phi(R_{s_j}, t_k) + \frac{\tilde{R}_j^k - R_{s_j}}{R_{s_j+1} - R_{s_j}}\phi(R_{s_j+1}, t_k). \tag{3.2.6}$$

Finally the discrete solution to the particle dissolution problem (3.1.2)-(3.1.3) is, for $j = 1, \ldots, m$,

$$n_{p,j}^{k+1}(\vec{x}) = \left(\frac{R_{s_j+1} - \tilde{R}_j^k}{R_{s_j+1} - R_{s_j}}\tilde{n}_{p,s_j}^{k+1}(\vec{x}) + \frac{\tilde{R}_j^k - R_{s_j}}{R_{s_j+1} - R_{s_j}}\tilde{n}_{p,s_j+1}^{k+1}(\vec{x})\right)\left(1-\Delta t\frac{\partial f}{\partial R}(R_j^{k+1}, c^k(\vec{x}))\right). \tag{3.2.7}$$

3.3 Discretization in space

The convection-diffusion equation (3.1.7) and the convection equation (3.1.1) are numerically solved by a stabilized Galerkin method on continuous piecewise linear finite elements. We use the SUPG stabilization method, which was introduced by Brooks and Hughes [8]. Let $h > 0$ and let $\mathcal{T}_h$ be a tetrahedral mesh of Ω_{el}, where all the tetrahedrons K_i of the mesh are of diameter $\mathrm{diam}(K_i) \le h$. Let V_h be the finite element space defined by

$$V_h = \{v \in C^0(\Omega_{\mathrm{el}}) : v|_K \in P_1(K), \forall K \in \mathcal{T}_h\}$$

where $P_1(K)$ is the space of all linear polynomials in tetrahedron K. We denote by $\bar{v} = \frac{1}{|\Omega_{\mathrm{el}}|}\int_{\Omega_{\mathrm{el}}} v d\Omega$ the mean of a function v over Ω_{el}. Suppose that $c_h^n \in V_h$ and $\tilde{n}_{p,j,h}^n \in V_h, j = 1, \ldots, m$, are known for $n = 0, \ldots, k$. Then c_h^{k+1} and $\tilde{n}_{p,j,h}^{k+1}$ belong to V_h and satisfy for all $v \in V_h$:

$$\begin{aligned}\int_{\Omega_{\mathrm{el}}} c_h^{k+1} v d\Omega + \frac{1}{2}\Delta t\int_{\Omega_{\mathrm{el}}} \vec{u}_h.\vec{\nabla} c_h^{k+1}(v-\bar{v})d\Omega - \frac{1}{2}\Delta t\int_{\Omega_{\mathrm{el}}} \vec{u}_h.\vec{\nabla} v(c_h^{k+1} - \bar{c}_h^{k+1})d\Omega \\ + D\Delta t\int_{\Omega_{\mathrm{el}}} \vec{\nabla} c_h^{k+1}.\vec{\nabla} v d\Omega + \beta_c\Delta t\sum_{K\in\mathcal{T}_h}\int_K \tau_K\frac{h_K}{2\|\vec{u}_h\|}(\vec{u}_h.\vec{\nabla} c_h^{k+1})(\vec{u}_h.\vec{\nabla} v)d\Omega \\ = \Delta t\int_{\Omega_{\mathrm{el}}} \dot{Q}_h^k v d\Omega + \int_{\Omega_{\mathrm{el}}} c_h^k v d\Omega + \beta_c\Delta t\sum_{K\in\mathcal{T}_h}\int_K \tau_K\frac{h_K}{2\|\vec{u}_h\|}\dot{Q}_h^k \vec{u}_h.\vec{\nabla} v d\Omega\end{aligned} \tag{3.3.1}$$

and

$$\begin{aligned}\int_{\Omega_{\mathrm{el}}} \tilde{n}^{k+1}_{p,j,h} v d\Omega + \frac{1}{2}\Delta t \int_{\Omega_{\mathrm{el}}} \vec{u}_h . \vec{\nabla} \tilde{n}^{k+1}_{p,j,h} (v - \bar{v}) d\Omega - \frac{1}{2}\Delta t \int_{\Omega_{\mathrm{el}}} \vec{u}_h . \vec{\nabla} v (\tilde{n}^{k+1}_{p,j,h} - \bar{\tilde{n}}^{k+1}_{p,j,h}) d\Omega \\ + \beta_{n_p} \Delta t \sum_{K \in \mathcal{T}_h} \int_K \frac{h_K}{2\|\vec{u}_h\|} (\vec{u}_h . \vec{\nabla} \tilde{n}^{k+1}_{p,j,h}) (\vec{u}_h . \vec{\nabla} v) d\Omega \\ = \Delta t \int_{\Omega_{\mathrm{el}}} S_{j,h} v d\Omega + \int_{\Omega_{\mathrm{el}}} n^k_{p,j,h} v d\Omega + \beta_{n_p} \Delta t \sum_{K \in \mathcal{T}_h} \int_K \frac{h_K}{2\|\vec{u}_h\|} S_{j,h} \vec{u}_h . \vec{\nabla} v d\Omega. \quad (3.3.2)\end{aligned}$$

The meshsize parameter h_K is defined by [35]

$$h_K = \frac{2\|\vec{u}_h\|}{\sum_{i=1}^4 |\vec{u}_h . \vec{\nabla} \phi_i(C_K)|}. \quad (3.3.3)$$

Here $\phi_1, \dots, \phi_4$ are the linear basis functions of K associated to the vertices, and C_K denotes the barycentre of K. The norm $\|.\|$ denotes the Euclidean norm.

The source term $\dot{Q}^k_h$ is approximated by

$$\dot{Q}^k_h(\vec{x}) \approx -4\pi \frac{\rho}{M} \sum_{i=1}^{m} n^{k+1}_{p,i,h}(\vec{x}) f(R_i, c^k(\vec{x})) R_i^2 \Delta R_i - \frac{I}{6FV}. \quad (3.3.4)$$

For the approximated feeding function $S^k_{j,h}$ we compute first the percentages of the mass to be added for each radius

$$p_j = \int_{R_{j-1}}^{R_j} \frac{1}{r\sigma\sqrt{2\pi}} \exp\left(-\frac{(\log r - \mu)^2}{2\sigma^2}\right) dr, \quad j = 1, \dots, m. \quad (3.3.5)$$

The integrals are approximated numerically. We compute $a_m = \sum_{j=1}^m p_j$. This quantity should be equal to one, but because we are working with a finite number of radii this is not the case and the mass will have to be corrected accordingly. As for the model we will correct the mass added to the bath at the end using constants $C_{s,j}$ (see below).

We approximate $S_1(\vec{x})$, defined in (1.2.8), by $S_{1,h}(\vec{x})$. The function $S_2(R)$ from (1.2.6) is approximated by piecewise constant functions $S_{2,j}$.

For each radius we compute the mass which we would add during Δt, using $S_{1,h}$ and $S_{2,j} \equiv 1$, giving

$$C'_{m,j} = \frac{4}{3}\pi\rho R_j^3 \Delta R_j \Delta t \int_{\Omega_{\mathrm{el}}} S_{1,h} d\Omega, \quad j = 1, \dots, m. \quad (3.3.6)$$

Remembering that the total mass we want to add is equal to C_m, the correction factor for each radius is

$$C_{S,j} = \frac{C_m p_j}{C'_{m,j} a_m}, \quad j = 1, \dots, m. \quad (3.3.7)$$

Finally the approximated feeding function is given by

$$S^k_{j,h} = \begin{cases} C_{S,j} S_{1,h}, & \text{if } t_k \in \{\tau^i\}, \\ 0, & \text{otherwise.} \end{cases} \quad (3.3.8)$$

We set $\tau_K = \max(0, 1 - 2D/(h_K\|\vec{u}\|))$. The parameters β_c and β_{n_p} are set equal to 1, a value which ensures the stability of the scheme.

Remark 3.3.1. We note here that the SUPG stabilization used is not strongly consistent, in the sense of [30]. This means that the solutions c and n_p of Problem 1.2.1 do not verify equations (3.3.1) - (3.3.2) because we omit the stabilization term

$$\beta_c \sum_{K\in\mathcal{T}_h} \int_K \tau_K \frac{h_K}{2\|\vec{u}\|}(c_h^{k+1} - c_h^k)(\vec{u}.\vec{\nabla} v)d\Omega \tag{3.3.9}$$

in equation (3.3.1) and the analogous term in (3.3.2). Bochev et al. [7] showed that the formulation including the term (3.3.9) remains well-posed for all timesteps. However, the stabilization we use is consistent in the sense that the order of convergence of the approximated solution to the correct solution is not changed by the stabilization. Hence, we will normally show results obtained without this term because they show less spurious oscillations.

The two problems (3.3.1) and (3.3.2) are discretized using Algorithm 2.9.3 described in Chapter 2, and the linear systems arising from the discretization are solved using GMRES algorithm [33].

Chapter 4

Numerical aspects

In the first section we discuss some details concerning the efficiency and accuracy of the numerical computations of the particle dissolution problem (3.1.2). We also give the solution of the dissolution problem if the dissolution kinetics is diffusion. Section 4.2 is devoted to the numerical aspects of the alumina consumption by electrolysis. We show and justify three different consumption models.

4.1 Particle dissolution

Mass conservation for the particle dissolution

We consider the following problem: Find c and n_p such that

$$\frac{\partial c}{\partial t} + \vec{u}.\vec{\nabla} c - D\Delta c = \dot{q}_1, \qquad \text{in } \Omega_{\text{el}} \times (0,T), \tag{4.1.1}$$

$$\frac{\partial c}{\partial n} = 0, \qquad \text{on } \partial\Omega_{\text{el}} \times (0,T), \tag{4.1.2}$$

$$\frac{\partial n_p}{\partial t} + \vec{u}.\vec{\nabla} n_p + \frac{\partial}{\partial R}(n_p f(R,c)) = 0, \qquad \text{in } \Omega_{\text{el}} \times (0,\infty) \times (0,T), \tag{4.1.3}$$

with initial conditions $c(\vec{x}, t = 0) = c^0(\vec{x})$ and $n_p(\vec{x}, R, t = 0) = n_p^0(\vec{x}, R)$. The dissolution is defined by

$$\dot{R}(t) = f(R(t), c(\vec{x}, t)), \tag{4.1.4}$$

and the source term for the convection-diffusion equation in c is given by

$$\dot{q}_1(\vec{x}, t) = -4\pi \frac{\rho}{M} \int_0^\infty n_p(\vec{x}, R, t) f(R, c(\vec{x}, t)) R^2 dR. \tag{4.1.5}$$

We showed in equation (1.2.30) that the mass lost by dissolution in (4.1.3) is equal to the mass of the source term of (4.1.1). This result implies that the mass of alumina is conserved.
The dissolution in equation (4.1.3) is discretized by (3.2.7). On the other hand, the source term of equation (4.1.1) is discretized by computing

$$\dot{q}_{1,h}^k(\vec{x}) \approx -4\pi \frac{\rho}{M} \sum_{i=1}^{m} n_{p,i,h}^{k+1}(\vec{x}) f(R_i, c^k(\vec{x})) R_i^2 \Delta R_i. \tag{4.1.6}$$

The mass balance is not exact. The mass lost or gained will tend to zero if Δt and ΔR_i tend to zero, but it is not equal to zero. Since the computational effort grows linearly with the number of radii used to discretize R, it would be advantageous to take a small number of discretization radii. In order to do this without having a bad mass balance we discretize $\dot{q}_1$ by

$$\dot{q}_{1,h}^{k}(\vec{x}) = -\frac{1}{\Delta t}\frac{4}{3}\pi\frac{\rho}{M}\sum_{i=1}^{m}\left(n_{p,i,h}^{k+1}(\vec{x}) - n_{p,i,h}^{k}(\vec{x})\right)R_i^3\Delta R_i, \tag{4.1.7}$$

i.e., we simply discretize the first term of equation (1.2.30) instead of the last term. Thus, the mass conservation is exact independently of the number of radii used for the discretization of R.

Efficiency and accuracy of the dissolution

The particle dissolution is described as

$$\frac{dR(t)}{dt} = f(R(t), c(\vec{x}, t)). \tag{4.1.8}$$

Particles are introduced at different times. For example, one particle population is introduced at time t_1 and another population at time t_2. It is computationally much easier if these populations can be merged instead of treating them separately. If, given a particle with size R at time t_3, we can compute the size at time $t_3 + \Delta t$ without knowing whether the particle has been introduced at time t_1 or t_2 we can combine the two populations. This property is satisfied if the function f, with c fixed, is bijective from $\mathbb{R}$ to $\mathbb{R}$. This seems to be a reasonable assumption because particles should dissolve and not get bigger.

But since the particle dissolution is an endothermic reaction, after the feeding to the bath the particles solidify the bath surrounding them and they grow in size. Their alumina mass does not change because the additional solid part is from the bath. For the computation of the dissolution we assume therefore that the particle size remains constant during a certain time, called latency time, and afterwards the particle size is strictly decreasing, following the equation (4.1.8). The algorithm computing the particle dissolution distinguishes therefore between the new particles, which do not change in size, and the old particles, which do dissolve. Each particle population which is younger than the latency time is kept separately from the others. Once a population is in the bath for more than the latency time it is added to the old particles which dissolve.

Computing the particle dissolution is essentially done using the method of characteristics. Thus, at each time step we make an error. Since the mass balance is exact, the error acts only on the dissolution time. The time of complete dissolution of a particle gets longer if we compute more time steps. We would therefore like to make as few timesteps as possible. On the other hand, for the convection-diffusion problem for the concentration we cannot take too large timesteps because the numerical diffusion becomes too important. We take therefore one timestep Δt_1 for the convection-diffusion problems and a second timestep $\Delta t_2 > \Delta t_1$ for the particle dissolution. The error for the total dissolution time does effectively diminish. Δt_2 can not be chosen too large because otherwise the particles move too far before they dissolve. Since the total dissolution time of a particle and the dissolution function f, which are obtained experimentally, are not precisely known, it is not very important to have the exact dissolution time in the simulation.

Discretization of the dissolution if the kinetics is diffusion

In Section 1.2 we introduced the dissolution law if dissolution kinetics is governed by diffusion. This law is given by

$$f(R, c) = -\frac{\alpha}{R}, \tag{4.1.9}$$

with $\alpha = DM(c_{\text{sat}} - c)/\rho$. To discretize the particle dissolution (3.1.2) in time using the above dissolution law we proceed as in Section 3.1. We fix $\vec{x} \in \Omega_{\text{el}}$ and we introduce, to simplify notation,

$$\phi(R,t) = n_p(\vec{x}, R, t),$$
$$g(R) = f(R, c^k(\vec{x})).$$

Equation (3.1.2) becomes

$$\frac{\partial \phi}{\partial t}(R,t) + \frac{\partial}{\partial R}(g(R)\phi(R,t)) = 0, \tag{4.1.10}$$

which we solve by the method of characteristics. Here we have

$$\dot{R}(t) = g(R(t)) = -\frac{\alpha^k}{R(t)}, \quad t \in [t_k, t_{k+1}], \tag{4.1.11}$$

which leads to $R(t) = \sqrt{\beta - 2\alpha^k t}$ with β a constant. We discretized here $\alpha^k = DM(c_{\text{sat}} - c^k(\vec{x}))/\rho$. Using the initial condition $R(t_k)$ we find $\beta = 2\alpha^k t_k + R^2(t_k)$. This gives

$$R(t_{k+1}) = \sqrt{2\alpha^k(t_k - t_{k+1}) + R^2(t_k)} = \sqrt{R^2(t_k) - 2\alpha^k \Delta t}, \tag{4.1.12}$$

or, written differently,

$$R(t_k) = \sqrt{R^2(t_{k+1}) + 2\alpha^k \Delta t}. \tag{4.1.13}$$

We set $\psi(t) = \phi(R(t), t)$. Denoting $\dot{\psi}(t) = \frac{d\psi(t)}{dt}$ we have

$$\dot{\psi}(t) = \frac{\partial \phi}{\partial t}(R(t),t) + \frac{\partial \phi}{\partial R}(R(t),t)\dot{R}(t) = \frac{\partial \phi}{\partial t}(R(t),t) + \frac{\partial \phi}{\partial R}(R(t),t) g(R(t)). \tag{4.1.14}$$

Using (4.1.10) and then the fact that $g(R(t)) = \dot{R}(t)$ we obtain

$$\dot{\psi}(t) = -\psi(t)\frac{dg}{dR}(R(t)) = \psi(t)\frac{\alpha^k}{R^2(t)} = -\psi(t)\frac{g(R(t))}{R(t)} = -\psi(t)\frac{\dot{R}(t)}{R(t)}, \tag{4.1.15}$$

and it is easy to verify that $\psi(t) = CR(t)$, C being a constant defined by

$$C = \frac{\psi(t_k)}{R(t_k)} = \frac{\phi(R(t_k), t_k)}{R(t_k)}. \tag{4.1.16}$$

So

$$\psi(t_{k+1}) = \frac{R(t_{k+1})}{R(t_k)}\phi(R(t_k), t_k), \tag{4.1.17}$$

and

$$\phi(R(t_{k+1}), t_{k+1}) = \frac{R(t_{k+1})}{\sqrt{R^2(t_{k+1}) + 2\alpha^k \Delta t}}\phi(R(t_k), t_k). \tag{4.1.18}$$

Computing the solution of the particle dissolution is thus explicit and we get

$$n_p^{k+1}(\vec{x}, R) = \frac{R(t_{k+1})}{\sqrt{R^2(t_{k+1}) + 2\alpha^k \Delta t}}\tilde{n}_p^{k+1}(\vec{x}, R). \tag{4.1.19}$$

For the discretization in R we proceed again in the same fashion as in Section 3.3. We introduce m radii $0 < R_1 < \ldots < R_m$. Suppose that the radius of a particle at position $\vec{x} \in \Omega_{\text{el}}$

and time t_{k+1} is R_j^{k+1}. From equation (4.1.13) we know that the radius of this particle at time t_k was

$$\tilde{R}_j^k = \sqrt{(R_j^{k+1})^2 + 2\alpha^k \Delta t}. \tag{4.1.20}$$

Let s_j be such that $R_{s_j} \leq \tilde{R}_j^k < R_{s_j+1}$. We interpolate to find

$$\hat{n}_{p,j}^{k+1}(\vec{x}) = \tilde{n}_{p,s_j}^{k+1}(\vec{x}) \frac{R_{s_j+1} - \tilde{R}_j^k}{R_{s_j+1} - R_{s_j}} + \tilde{n}_{p,s_j+1}^{k+1}(\vec{x}) \frac{\tilde{R}_j^k - R_{s_j}}{R_{s_j+1} - R_{s_j}}. \tag{4.1.21}$$

Using (4.1.19) we compute the particle density at time t_{k+1} by

$$n_{p,j}^{k+1}(\vec{x}) = \frac{R_j}{\sqrt{R_j^2 + 2\alpha^k \Delta t}} \hat{n}_{p,j}^{k+1}(\vec{x}), \quad j = 1, \ldots, m. \tag{4.1.22}$$

4.2 Liquid alumina consumption

Electrolysis takes place at the interface between the anode and the electrolyte and at the interface between the cathode and the electrolyte. Our first idea was therefore to model liquid alumina consumption as a boundary outflow condition, setting

$$D \frac{\partial c}{\partial n} = -\kappa \tag{4.2.1}$$

at the boundary which forms the interface anode - bath, with

$$-\kappa = -\frac{1}{6} \frac{\vec{j}.\vec{n}}{F}. \tag{4.2.2}$$

We recall that $\vec{j}$ stands for the current density and F is the Faraday constant. The problem of this formulation is that the concentration gradient, which is going to appear at the anode - bath interface, will be concentrated on a length of about 10^{-9} meters, due to the diffusion coefficient. This means that in order to resolve the gradient a mesh of meshsize 10^{-9} near this interface has to be used. The generation of such a mesh, as well as the computations on it, would be complicated. We did not attempt to work with such a mesh.

Simple spatial model

Instead we use much simpler models where consumption takes place nearly everywhere in the bath. This is justified by the fact that chemical reactions which are part of the overall transformation of alumina to aluminium take place in the whole bath. One possibility would be to make a numerical model of the chemical reactions. We will not do this for two reasons. First, such models tend to be rather complicated, and since there are several chemical reactions to model, the computational cost to solve all these equations would be very high compared to the time spent solving the more inportant rest of the alumina model. Secondly, the precise chemical reactions in the bath are not known. We could still set up some simplified chain of reactions, but then the result would not be precise anyway and thus we can as well use a simpler consumption model.

In the simplest consumption model we compute the total decrease of alumina concentration in the bath per time unit, divide this result by the volume of the bath and subtract this quantity of the alumina concentration in the whole bath. To make the model a bit more realistic we suppose that the consumption takes place only between the bottom of the anodes and the interface between the bath and the metal pad. We get the model that we described in Chapter 1,

$$\dot{q}_2 = -\frac{I}{6FV}, \tag{4.2.3}$$

(see (1.2.17)).

Threshold model

Now, depending on the velocity field, it could happen that some parts of the bath are underfed with alumina, i.e., the concentration increase due to dissolving particles and concentration transport is smaller than the decrease due to the electrolysis model. This implies that in such parts the concentration decreases over time, reaching negative values, which are not physically sound. In reality this can certainly not happen, but if the alumina concentration at some point of the bath becomes close to zero, the electric current will rather pass through some other part of the bath with a higher alumina concentration, because the electrical resistance there is lower. Thus, in a second model we determine at each time step the domain where the concentration is higher than a certain value $\epsilon > 0$,

$$\Omega_{\text{el}}^{k,\epsilon} = \{\vec{x} \in \Omega_{\text{el}},\ \text{between the bottom of the anodes and } \Gamma : c^k(\vec{x}) > \epsilon\}. \tag{4.2.4}$$

Then we set $V^{k,\epsilon} = |\Omega_{\text{el}}^{k,\epsilon}|$ and

$$\dot{q}_2^k(\vec{x}) = \begin{cases} -\frac{I}{6FV^{k,\epsilon}}, & \vec{x} \in \Omega_{\text{el}}^{k,\epsilon}, \\ 0, & \text{otherwise.} \end{cases} \tag{4.2.5}$$

In this way the concentration will always remain greater than ϵ.

Remark 4.2.1. We note that if the alumina concentration does not go below the threshold ϵ, the two consumption models described so far are equivalent. However, if we do not know if the velocity field will convect the concentration well over the whole bath or not it is safer to use the second model.

Current density model

In an apparently even more realistic model we have an alumina consumption which depends on the current density $\vec{j}$. We say apparently, because there are two problems. The first problem is related to the physical meaning of the current density. If we take any surface Γ_{el} which stretches over the whole bath, we have that

$$I = \int_{\Gamma_{\text{el}}} \vec{j}.d\vec{\sigma}. \tag{4.2.6}$$

Hence the use of $\vec{j}$ implies that the consumption should take place on a surface. But as a boundary conditions this is not possible, for the reasons cited above, and we do not know where to choose a surface in the bath. Therefore we decide to use a spatial consumption model. Thus we have to transform the current density to some scalar value.

The second problem is related to the values of the current density that come from a numerical computation. The current density depends on the distance between the anodes and the metal pad. This distance changes due to the wear of the anodes, and the computations done to obtain the current density do not take this effect into account. Thus, the current density $\vec{j}_h$ we use is a rather rough approximation of the real current density $\vec{j}$.

For this model we suppose again that the consumption takes place only if the concentration is greater than some value ϵ. To transform the vectorial $\vec{j}_h$ to a scalar we first compute at each time step a total current density in the bath,

$$|\vec{j}_h^k| = \int_{\Omega_{\mathrm{el}}^{k,\epsilon}} \vec{j}_h . \vec{j}_h d\Omega, \tag{4.2.7}$$

and we set the local current density equal to $I(\vec{j}_h(\vec{x}).\vec{j}_h(\vec{x}))/|\vec{j}_h^k|$, which ensures that the integral of the local density over the part of the bath where the consumption takes place is equal to I. Then

$$\dot{q}_2^k(\vec{x}) = \begin{cases} -\frac{I\vec{j}_h(\vec{x}).\vec{j}_h(\vec{x})}{6FV^{k,\epsilon}|\vec{j}_h^k|}, & \vec{x} \in \Omega_{\mathrm{el}}^{k,\epsilon} \\ 0, & \text{otherwise.} \end{cases} \tag{4.2.8}$$

It is difficult to judge whether this consumption model is better than the two previous or not. In the following we will in general work with the threshold model, even if normally the concentrations do not get close to zero.

Chapter 5

Numerical Results

Here we present some numerical results obtained using the model, the numerical schemes and the discretization of the first chapters. First we present the convergence of an academic convection problem with known solution, to show that our code attains optimal convergence. In Section 5.2 we numerically compare the different numerical schemes that were presented in Chapter 2. With the best scheme, which is the scheme presented in Section 2.9, we then compute solutions to the complete model of Chapter 1. We do this for different velocity fields, and we show that the numerical solution becomes periodic in time.

5.1 Convergence test with known solution

We consider the convection problem with Dirichlet boundary conditions on the incoming boundary, where $\vec{u}.\vec{n} < 0$, which we denote by Γ_-. Our problem is thus: Find $c = c(\vec{x}, t)$ such that

$$\frac{\partial c}{\partial t} + \vec{u}.\vec{\nabla} c = 0, \qquad \text{in } \Omega \times (0,T), \tag{5.1.1}$$

$$c = 0, \qquad \text{on } \Gamma_- \times (0,T), \tag{5.1.2}$$

$$c = c^0, \qquad \text{in } \Omega \times \{0\}. \tag{5.1.3}$$

For this test we use a consistent stabilization in the discretization, i.e., the stabilization term is

$$\beta_c \Delta t \sum_K \int_K \frac{h_K}{\|\vec{u}\|} \left[\frac{\partial c}{\partial t}(\vec{u}.\vec{\nabla} v) + (\vec{u}.\vec{\nabla} c)(\vec{u}.\vec{\nabla} v) \right] d\Omega. \tag{5.1.4}$$

We set $\Omega = [-1,1]^3$, $\vec{u} = (-\frac{3}{\pi} y, \frac{3}{\pi} x, 0)$,

$$c_0 = \begin{cases} 200 \exp\left(\dfrac{-0.6}{0.3 - (x-0.5)^2 - y^2 - z^2} \right), & \text{if } (x-0.5)^2 + y^2 + z^2 < 0.3, \\ 0, & \text{if } (x-0.5)^2 + y^2 + z^2 \geq 0.3, \end{cases} \tag{5.1.5}$$

$\beta_c = 1$. We compute the relative error in the L^2-norm after $T = 6$ seconds, i.e., we compute

$$e_c := \frac{\|c(T) - c_h(T)\|_{L^2(\Omega)}}{\|c(T)\|_{L^2(\Omega)}}. \tag{5.1.6}$$

Here the exact solution is equal to the initial condition, i.e., $c(T) = c^0$. We know that convergence of the approximated solution c_h to the exact solution c should be in $\mathcal{O}(h^{3/2} + \Delta t)$ (see [10]). On Figure 5.1 we show the relative error for $h = 1/16, 1/32, 1/64, 1/128$, with $\Delta t = 1/64, 1/181, 1/512, 1/1449$, hence $\Delta t \approx h^{3/2}$. The relative error, shown as a function of h on the figure, decreases in $h^{3/2}$. We have thus numerically verified the optimal convergence of the approximation to the solution.

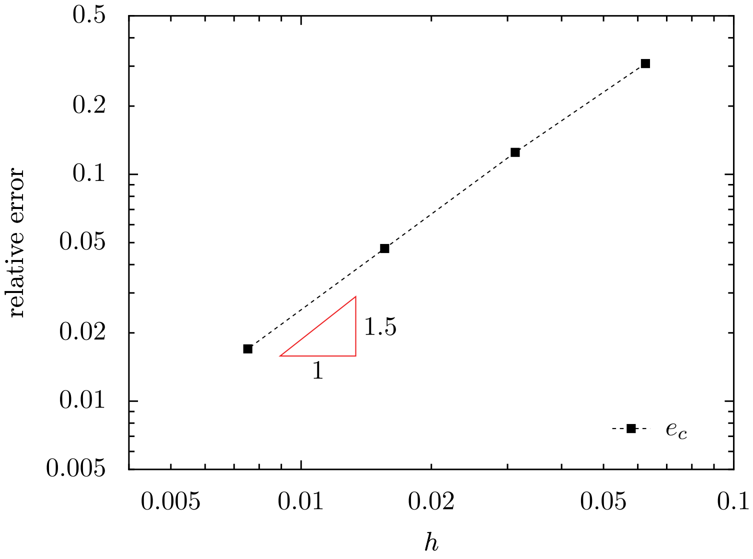


Figure 5.1: The relative error e_c with respect to h. We have $\Delta t \approx h^{3/2}$.

5.2 Comparison of different weak formulations of the convection-diffusion equation

In this section we will compare the different numerical schemes described in Chapter 2. The source term of the convection-diffusion problem is set to zero, and since the diffusion is reducing the differences between the results we omit it here. Hence, we consider the following problem: Find the solution $c = c(\vec{x}, t)$ of the problem

$$\frac{\partial c}{\partial t} + \vec{u}.\vec{\nabla} c = 0, \qquad \text{in } \Omega \times (0,T), \tag{5.2.1}$$

$$\frac{\partial c}{\partial n} = 0, \qquad \text{on } \partial\Omega \times (0,T), \tag{5.2.2}$$

$$c = c^0, \qquad \text{in } \Omega \times \{0\}. \tag{5.2.3}$$

Here we take $\vec{u}(x_1, x_2, x_3) = (-\cos(\frac{\pi}{2}x_1)\sin(\frac{\pi}{2}x_2), \sin(\frac{\pi}{2}x_1)\cos(\frac{\pi}{2}x_2), 0)$ and $\Omega = [-1,1] \times [-1,1] \times [-0.1, 0, 1]$. The final time T will be specified below.

The velocity field $\vec{u}_h$ we actually use is an approximation of $\vec{u}$, obtained by solving problem 1.3.2, and hence it is such that $\mathrm{div}(\vec{u}_h)$ in Ω and $\vec{u}_h.\vec{n}$ on $\partial\Omega$ are small, but not equal to zero.

We state the three properties which the approximation c_h should have:

$$\frac{d}{dt}\int_\Omega c_h d\Omega = 0, \tag{5.2.4}$$

$$\text{if } c^0 = \text{constant, then } c_h = c^0, \tag{5.2.5}$$

$$\frac{d}{dt}\|c_h(t)\|_{L^2(\Omega)} \leq 0, \forall t > 0. \tag{5.2.6}$$

We introduce the following quantities:

- The mass variation, denoted by $\mathrm{V}_{\Delta t}$ and computed as

$$\mathrm{V}_{\Delta t}(c_h(t)) := \sum_{i=1}^{M} \frac{\left|\int_\Omega c_h(t_i)(\vec{x})d\Omega - \int_\Omega c_h(t_{i-1})(\vec{x})d\Omega\right|}{\int_\Omega c^0(\vec{x})d\Omega}, \tag{5.2.7}$$

where $t = M\Delta t$ and $t_i = i\Delta t$. If the condition (5.2.4) is true, then the mass variation is zero.

- The variance, denoted by Var and computed as

$$\mathrm{Var}(c_h(t)) := \frac{1}{t}\sum_{i=1}^{M} \frac{\int_\Omega (c_h(t_i)(\vec{x}) - \bar{c}_h(t_i))^2 d\Omega}{\int_\Omega d\Omega}, \tag{5.2.8}$$

where $\bar{c}_h(t) = \frac{1}{|\Omega|}\int_\Omega c_h(t)(\vec{x})d\Omega$. If condition (5.2.5) is true, then the variance is zero.

- The positive L^2-variation, denoted by $\mathrm{V}_{\Delta t}^{L^2+}$ and computed as

$$\mathrm{V}_{\Delta t}^{L^2+}(c_h(t)) = \sum_{i=1}^{M} \frac{\max\{0, \|c_h(t_i)\|_{L^2(\Omega)} - \|c_h(t_{i-1})\|_{L^2(\Omega)}\}}{\int_\Omega c^0(\vec{x})d\Omega}. \tag{5.2.9}$$

If the condition (5.2.6) is true, then the positive L^2-variation is zero.

In order to check the three properties of the solution of each scheme presented in Section 2 we solve the problem (5.2.1) - (5.2.3) with two different initial conditions. In the first case we start with $c^0 = 1$, and we look if the variance of c_h is small. In the second case we start with

$$c^0(\vec{x}) = \begin{cases} 200e^{-0.02/(0.01-(x_1-x_1^0)^2-(x_2-x_2^0)^2)}, & \text{if } (x_1 - x_1^0)^2 + (x_2 - x_2^0)^2 < 0.01, \\ 0, & \text{if } (x_1 - x_1^0)^2 + (x_2 - x_2^0)^2 \geq 0.01. \end{cases} \tag{5.2.10}$$

Here we look at the mass variation and at the positive L^2-variation of the solution. We will compute the solution of each case on a regular, nonstructured grid of the domain Ω. The final time for the first case is $T = 100$ seconds, for the second case it is $T = 10000$ seconds. In addition, for the schemes which use a modified velocity field we compute the relative difference between the modified velocity field, which is always a projection $\Pi\vec{u}_h$, and $\vec{u}_h$,

$$e_{\vec{u}_h} = \frac{\|\vec{u}_h - \Pi\vec{u}_h\|_{L^2(\Omega)}}{\|\vec{u}_h\|_{L^2(\Omega)}}. \tag{5.2.11}$$

We label the different schemes according to Table 5.1 to refer to them more easily. The results for the different schemes are given in Table 5.2. In order to discuss the results we first look at the variance of the numerical solution. Here for most of the schemes the solution is equal to the constant initial condition. As predicted theoretically only schemes 2 and 3 do not conserve the trivial solution.

For the positive L^2-variance we also get the result we expected, i.e., the schemes 3, 4, 9 and 10 are L^2-stable and all the other schemes are not. We note however that especially the solution of scheme 5 is really not L^2-stable, and also the direct formulation 1 does not do very well.

Maybe the most surprising result is the mass variation. The schemes which were shown theoretically to not conserve mass, 1, 3 and 5, clearly do not conserve the mass. But the other schemes, which should conserve the mass do not have zero mass variation either. Especially the projected velocity field on sine and cosine functions (scheme 4) seems not to be that useful. For the other schemes the mass variation is 10^{-7} or smaller, which is maybe not what we expected but considering the final time of $T = 10000$ seconds it is nevertheless a good result. Since we are going to do simulations with a final time of this order the precision we attain is sufficient.

Label	Short description	Section
1	direct weak formulation	2.2
2	simple mass conserving formulation	2.3
3	simple L^2-stable formulation	2.4
4	velocity field projected on sinus- and cosinus-functions	2.5
5	velocity field projected on divergence free finite elements	2.6
6	approximated Helmholtz decomposition of the velocity field	2.7
7	meanfree mass conserving formulation, for GMRES	2.8
8	meanfree mass conserving formulation, general	2.8
9	meanfree L^2-stable mass conserving formulation, for GMRES	2.9
10	meanfree L^2-stable mass conserving formulation, general	2.9

Table 5.1: Labels for different schemes to be used for the presentation of the results

scheme	$\mathrm{V}_{\Delta t}(c_h(10000))$	$\mathrm{Var}(c_h(100))$	$\mathrm{V}_{\Delta t}^{L^2+}(c_h(10000))$	$e_{\vec{u}_h}$
1	0.11	0	0.013	0
2	$3.4{\cdot}10^{-7}$	$1.7{\cdot}10^{-3}$	$3.1{\cdot}10^{-3}$	0
3	0.076	$4.2{\cdot}10^{-4}$	0	0
4	$2.1{\cdot}10^{-4}$	0	0	$6.6{\cdot}10^{-6}$
5	2.1	0	35	$6.1{\cdot}10^{-7}$
6	$2.7{\cdot}10^{-7}$	0	$4.3{\cdot}10^{-5}$	$2.3{\cdot}10^{-7}$
7	$2.7{\cdot}10^{-7}$	0	$1.2{\cdot}10^{-3}$	0
8	$3.5{\cdot}10^{-10}$	0	$1.2{\cdot}10^{-3}$	0
9	$8.8{\cdot}10^{-8}$	0	0	0
10	$8.1{\cdot}10^{-9}$	0	0	0

Table 5.2: Comparison of the different numerical schemes presented in Chapter 2.

In conclusion we repeat what we already found after the theoretical considerations in Chapter 2, that the best schemes are the two last ones. Remembering that scheme 10 leads to matrices which are sparse and the associated linear system can be solved by general linear solvers, also direct ones, whereas this is not the case for the scheme 9, we conclude that the most appropriate scheme to our ends is scheme 10.

5.3 Simulation of a real case

In this section we want to do a simulation of the complete alumina problem defined in Chapter 1 in a real aluminium electrolysis cell. We want to see how the alumina concentration changes in the long run, and we verify if the numerical cell behaves qualitatively like a real cell. In particular we want to reach a state of the cell when the concentration evolution is periodic in time. This behaviour is observed in real cells, and we will explain it in the following. We then give the setting of the simulation and introduce the different velocity fields that we will use. Numerical results are presented afterwards, and we end the section by concluding remarks. This section is the basis for the validation presented in the next chapter.

Periodic behaviour of the concentration in the bath

In our model of Chapter 1 we assume that the current efficiency is constant, and therefore alumina consumption by electrolysis is constant over time. Since we know this, we add alumina quantities which ensure that the mass of alumina in the bath at the beginning and at the end of a feeding cycle is the same.

We already pointed out in the introduction that this is not possible in a real aluminium cell since the current efficiency, and thus the quantity of alumina consumed by electrolysis, is not known. In a real cell the bath is either under- or overfed, i.e., either the quantity of alumina added during one feeding cycle is smaller than the quantity consumed by electrolysis, or it is larger. Without going into details, the cell is underfed until some measurements of the electric resistance indicate that the alumina concentration in the bath is too low. Then the cell is overfed, again until one notices indirectly that the alumina concentration is starting to get too high. The reason for this under- and overfeeding is that the measurements of the resistance can be done more precisely when the alumina concentration is changing than if it remains almost constant. Hence, the cell is underfed and then overfed, always during several feeding cycles, the number of cycles being variable. Engineers know that after about 10 minutes, during which the cell has been run with the same feeding mode, the cell state becomes almost periodic. By periodic we mean that the alumina concentration at some point in the bath is the same at time t and at time $t + \Delta T$. The cell state cannot become really periodic, since either the alumina concentration is increasing or decreasing in the long run due to the over- or underfeeding. But apart from this decrease or increase, there is no apparent change in the alumina distribution in the bath (also this information has not been found out by measuring the alumina concentration directly, but comes indirectly from observations of for example the current efficiency or the current distribution).

We are led to the conclusion that we should reach a periodic state in our computations because we ensure that the alumina concentration in the bath is constant. The question is obviously whether this is true with our model. We will not do a theoretical research on this subject, which would consist in proving that our model admits periodic solutions. But we will do long simulations of our complete model in order to see whether some periodic solution appears or not. We note here that a periodic state of the cell is only possible because we ensure mass conservation. Without this property we would never reach periodicity.

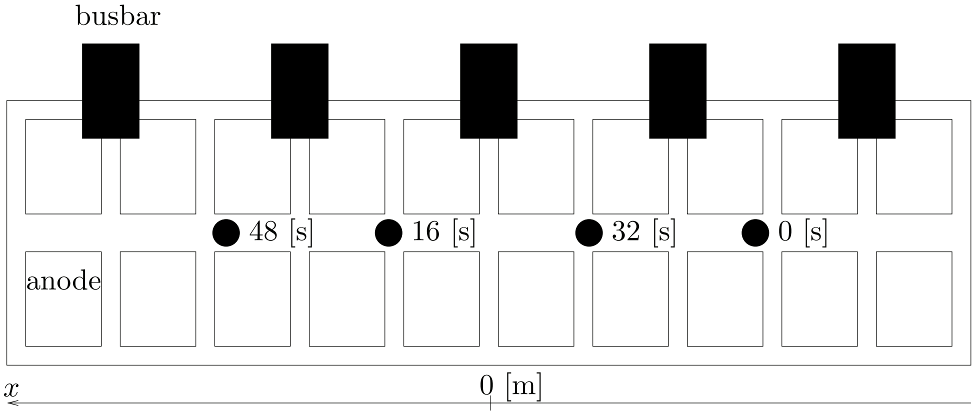


Figure 5.2: Schematic representation of the aluminium cell seen from above, with the feeders and the feeding times of each feeder during one feeding cycle.

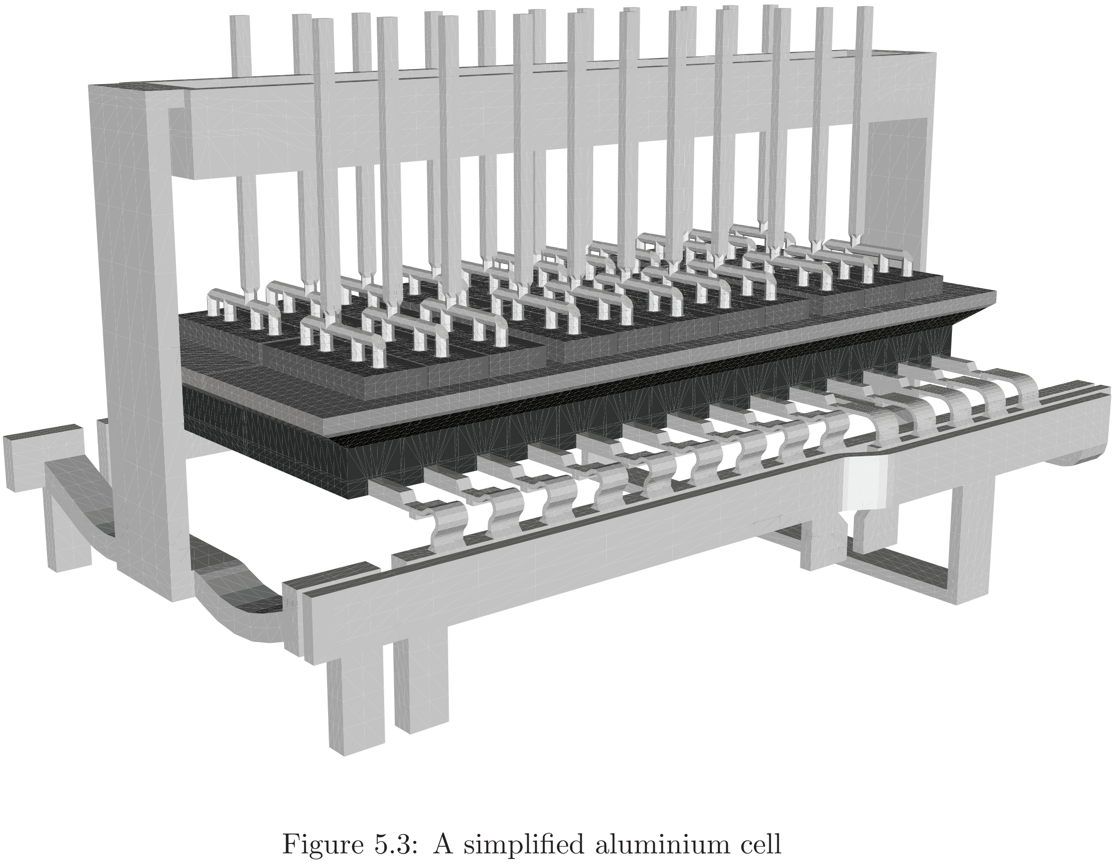


Figure 5.3: A simplified aluminium cell

Setting of the simulation

Hence, what we do here is simulating an aluminium pot over a long time. We start with a completely uniform concentration of 3 [wt%] in the whole bath. We set the feeding cycle length to 64 seconds. There are 4 alumina feeders, at x-positions -4.4, -1.6, 1.6, 4.4 [m], where we have set the zero of the x-axis in the middle of the cell. The alumina load of each feeder is approximately 0.9 [kg]. The order of the feeders is -4.4, 1.6, -1.6, 4.4 [m], adding their load to the bath in each feeding cycle at times 0, 16, 32, 48 [s] respectively. A schematic representation of the aluminium cell, showing the feeder positions and their feeding time during one feeding cycle is shown on Figure 5.2. On Figure 5.3 we show a simplified aluminium cell to give an overall picture of the situation. There we see electrical conductors around the cell in light grey. Of the parts which are not shown in the lightest grey tone, we have from top to bottom the anodes, the bath, the metal pad (in black) and the cathode. The feeders are not shown. The cell shown is different from the one we use for our computations.

We note that for all the computations done hereafter we multiply the particle dissolution curve shown on Figure 1.2 by a factor 10/6. Thus, the dissolution time of the biggest particles is now 10 seconds. This dissolution time is more adapted for our computations, on the basis of a research done by Groulier [21].

Velocity fields used for the simulation

We do this simulation for three different velocity fields. We recall that in Section 1.3 we had some boundary conditions on $\vec{u}$, especially $\vec{u} = 0$ on $\partial\Omega$, and $\vec{u}.\vec{n} = 0$ on Γ, where Ω is the domain of the fluids and Γ is the interface between the bath and the metal pad. We do not touch the condition on the interface, but we change the other condition to the following boundary conditions, which will lead to three different velocity fields:

$\vec{u}_a$: $\vec{u} = 0$ on $\partial\Omega$,

$\vec{u}_b$: $\vec{u} = 0$ on $\partial\Omega$, except at the bottom of the anodes, where we impose $\vec{u}.\vec{n} = 0$,

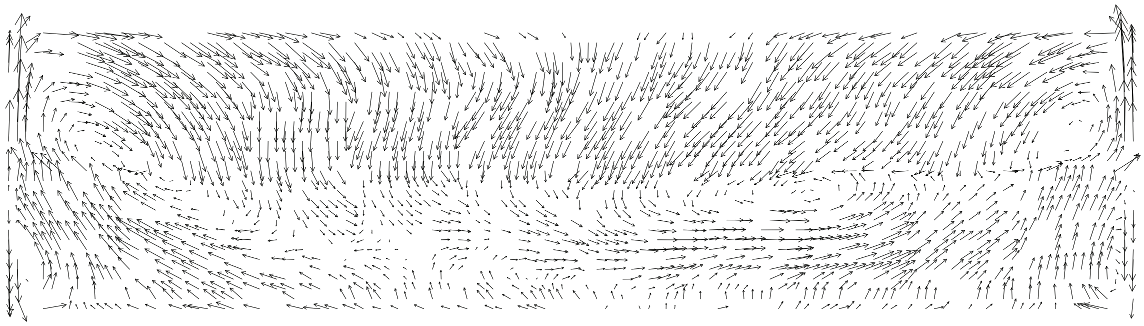
$\vec{u}_c$: $\vec{u} = 0$ on $\partial\Omega$, except at the exterior boundary of the cell, where we impose $\vec{u}.\vec{n} = 0$ (in all corners we numerically set $\vec{u} = 0$ since we do not have a normal at these points).

All these velocity fields are computed by solving problem 1.3.4, i.e., we obtain three different $\vec{u}_h^{\text{bubble}}$. However, we remove the bubble functions from $\vec{u}_h$, because the validation of the next chapter is more accurate without the bubble functions.

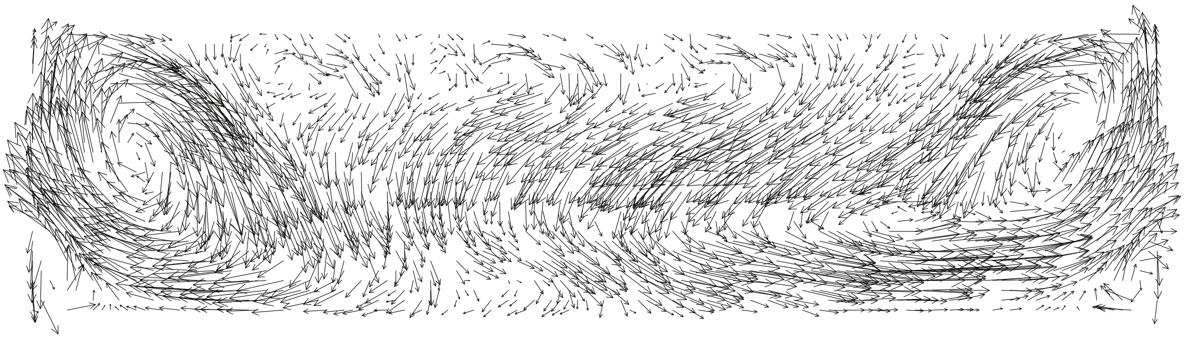
In the third velocity field we will have a faster moving liquid close to the boundary of the bath, which could considerably change the way the concentration is distributed. The velocities are higher when we use velocity field $\vec{u}_b$. We therefore expect a better stirring of the alumina concentration, and thus globally more uniform concentration distributions.

We also computed the veolcity field with $\vec{u}.\vec{n} = 0$ on the whole boundary $\partial\Omega$, but it turned out that the velocities became much higher than with the other boundary conditions. Since the velocities for the other boundary conditions were closer to velocity measurements we will not use the one obtained by imposing $\vec{u}.\vec{n} = 0$ on the whole boundary $\partial\Omega$.

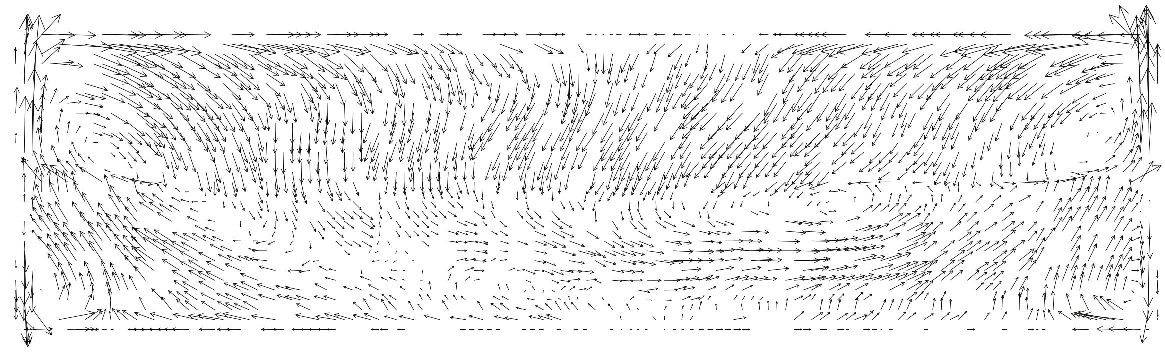
We show the nodal velocities at the interface Γ on Figure 5.4. First we remark that the last velocity field has velocities on two more nodes in each direction. This is because on the other velocity fields we imposed $\vec{u} = 0$ on the outer boundary, and hence these nodes do not appear on the figure. Otherwise we note that the general flow field of the three velocity fields is the same, with $\vec{u}_b$ being the fastest, which is not surprising since the bath is not very deep in z-direction and hence the influence of the boundary condition at the bottom of the anodes has a big influence on the speed of the velocities. The last velocity field moves faster close to the boundary, which is exactly what we expected. The maximum speeds are about 8 [cm/s] for $\vec{u}_a$, 11 [cm/s] for $\vec{u}_b$, and 10 [cm/s] for $\vec{u}_c$. The maximum for $\vec{u}_c$ is only reached close to the corners, otherwise the speed of $\vec{u}_c$ is very similar to $\vec{u}_a$. We notice that there are a lot of small vortices in all the velocity fields. The maximal velocity speed in z-direction is about 1.5 [cm/s] for all velocity fields, which is very fast, in comparison with the height of the bath of less than 4 [cm].



$\vec{u}_a$: $\vec{u} = 0$ on $\partial\Omega$



$\vec{u}_b$: $\vec{u} = 0$ on $\partial\Omega$, $\vec{u}.\vec{n} = 0$ at the bottom of the anodes



$\vec{u}_c$: $\vec{u} = 0$ on $\partial\Omega$, $\vec{u}.\vec{n} = 0$ at the exterior boundary

Figure 5.4: The different velocity fields, shown at the bath - aluminium interface, seen from above. The scale of the velocities is the same for the three images.

Results

In order to show the evolution of the alumina concentration in the bath we first show the variance $\mathrm{Var}(c(t))$ over time on Figures 5.5 - 5.7. Observe that the scales of the graphs are not the same for all the figures. We see that the variance increases rapidly at the beginning. At the end the values of the variance are always contained in a fixed interval. For the velocity field $\vec{u}_a$ this convergence of the interval takes about 4000 seconds, whereas for the velocity fields $\vec{u}_b$ and $\vec{u}_c$ the final interval is already reached after 2000 seconds. The variance of the concentration is much smaller using $\vec{u}_b$ than using the two other velocity fields. Hence, the concentration is much better distributed over the bath using $\vec{u}_b$. Since the bath moves faster for this velocity field this result is not very surprising.

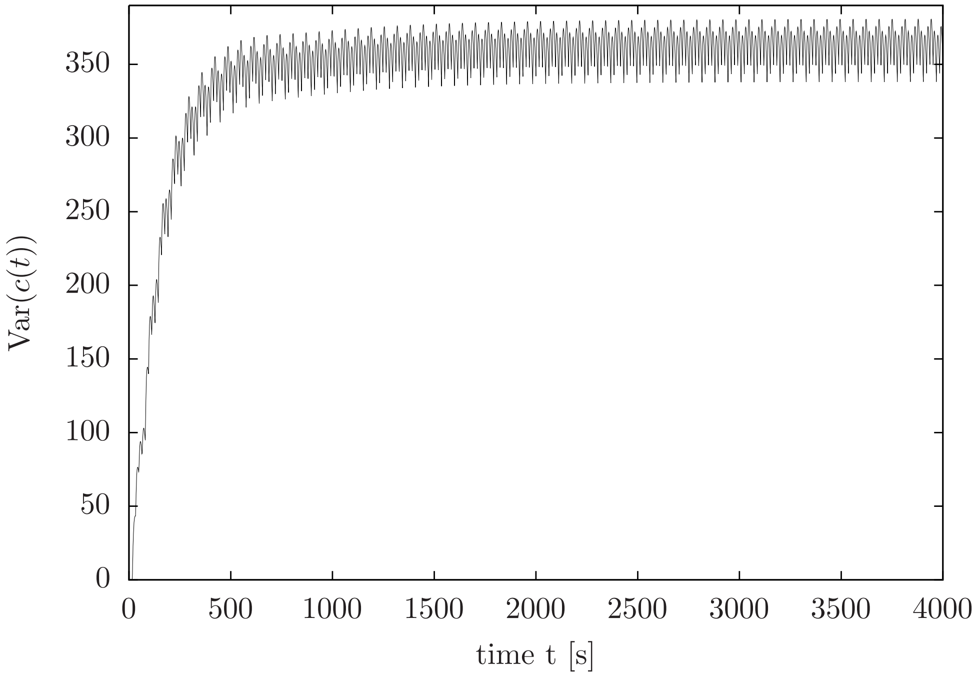


Figure 5.5: The variance of the alumina concentration in the bath as a function of time, for velocity field $\vec{u}_a$.

Since we expect the alumina concentration in the bath to become periodic in time, we show on Figure 5.8 the variance of $c(t)$ for all three velocity fields at multiples of the feeding period, i.e., for times $t = k\Delta T, k = 0, 1, \ldots$, with $\Delta T = 64[s]$ the feeding period. We notice that the variance $\mathrm{Var}(c(k\Delta T))$ converges to some value depending on the velocity field. Hence, at least the variance of the concentration starts to get periodic, with a period of at most ΔT. We also see that the form of the two variance curves obtained using $\vec{u}_a$ and $\vec{u}_c$ is the same: first, the variance increases rapidly, and after about 300-500 seconds it starts to converge slowly to the final value. The fact that the variance of the concentration using velocity field $\vec{u}_a$ is slightly higher than using velocity field $\vec{u}_c$ can also be explained by the faster moving velocity $\vec{u}_c$.

On Figure 5.9 we show the concentration variance during one feeding period towards the final time of the simulation. The variance curves are on different graphs because the y-scales are very different. Here we see that the periodicity of the variance is exactly ΔT, and each addition of alumina particles to the bath is clearly visible on the variance plot.

Finally we show the concentration at the interface between the bath and the metal pad and

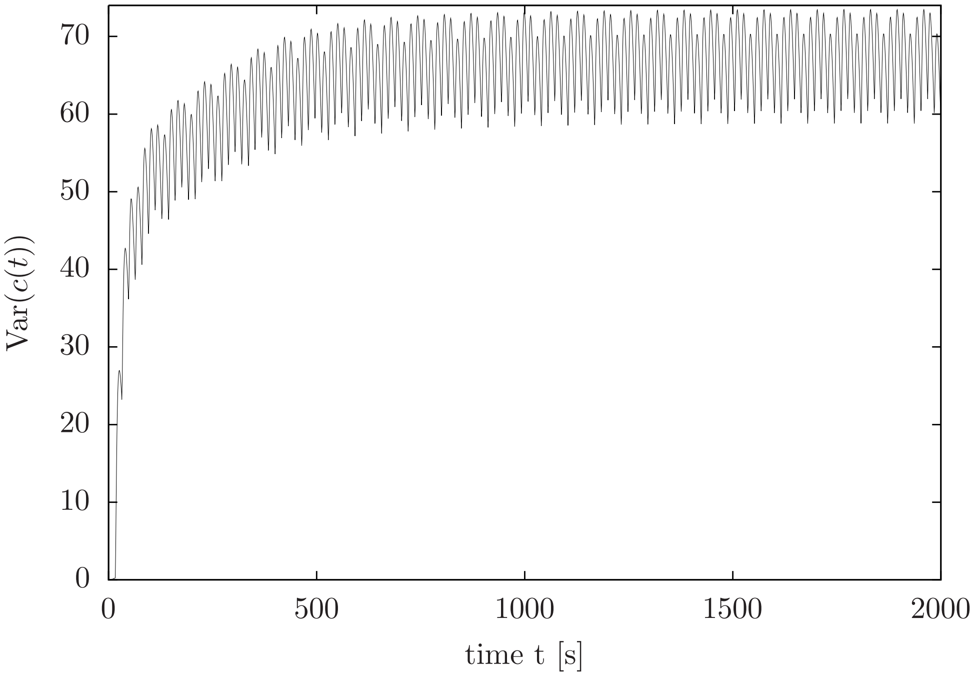


Figure 5.6: The variance of the alumina concentration in the bath as a function of time, for velocity field $\vec{u}_b$.

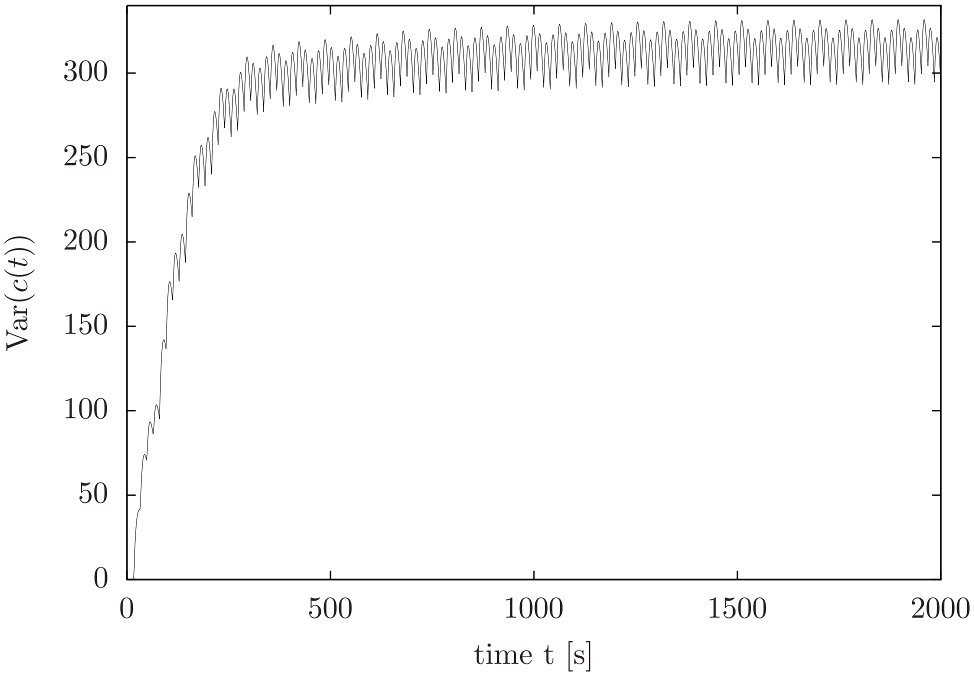


Figure 5.7: The variance of the alumina concentration in the bath as a function of time, for velocity field $\vec{u}_c$.

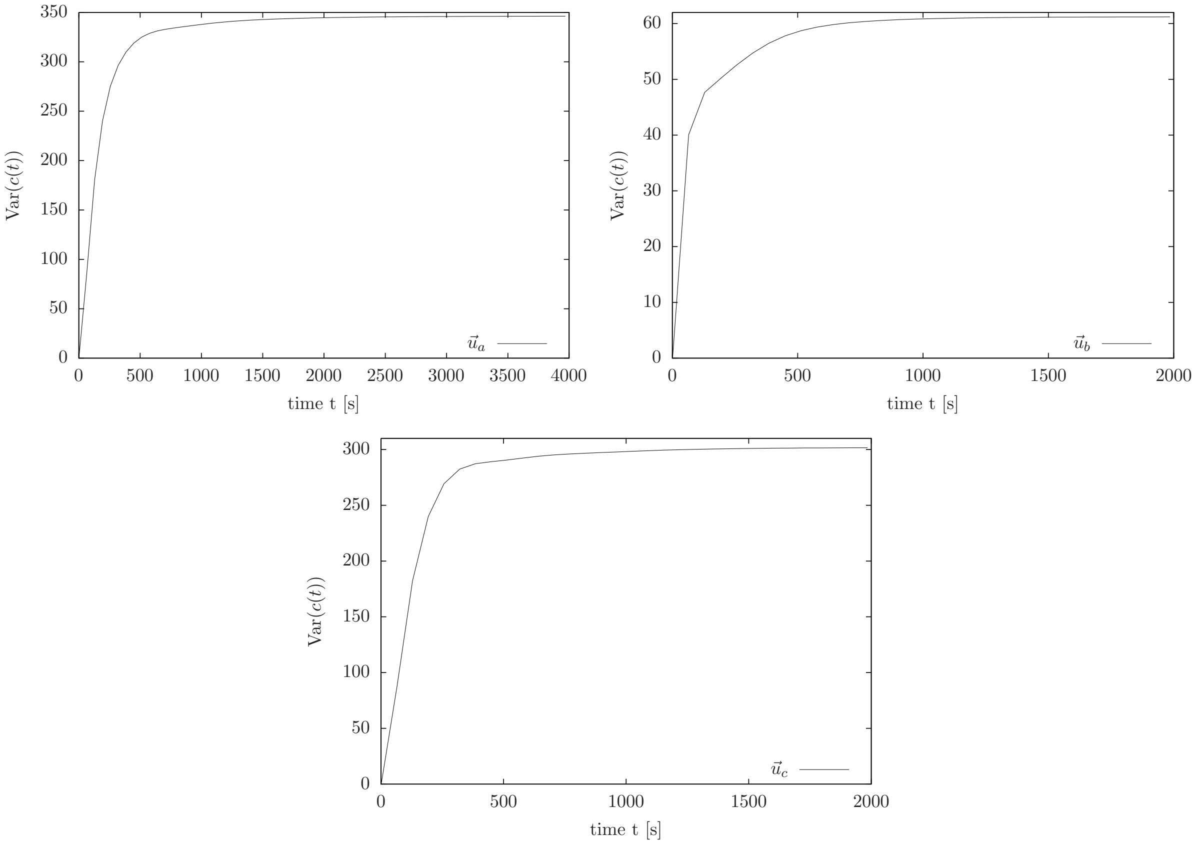


Figure 5.8: The variance of the alumina concentration in the bath at multiples of the feeding period, for the different velocity fields.

at the time $t = 5000$ seconds for the three different velocity fields. On Figure 5.10 we take the linear scale min($c(t)$) - max($c(t)$) for the concentration obtained using velocity field $\vec{u}_b$, and on Figure 5.11 we use the linear scale bound by the minimum and maximum of the concentration computed using velocity field $\vec{u}_a$. We see that the concentrations for velocity fields $\vec{u}_a$ and $\vec{u}_c$ are very similar, and also the concentration for velocity field $\vec{u}_b$ shows similarities to the other results, only that here the concentration is much better distributed.

Conclusion

We conclude that the concentration distribution reaches a periodic state, and this final state depends very much on the underlying velocity field. According to engineers the periodic state of a real aluminium cell is reached in about 10 minutes. Looking again at Figure 5.8 we see that the variance has almost converged after about 10 minutes for all the velocity fields. The combination of the velocity fields and the model seems thus very reasonable from this point of view.

Another question is now if the final periodic state is independent of the initial condition. This is certainly not the case if we allow any initial condition, but if we fix the initial alumina quantity in the bath and we vary only the initial distribution of the concentration, do we always converge to the same periodic solution? Again we will not study this question theoretically, but our computational results indicate that the answer to this question is yes.

These two properties, the convergence to a periodic solution and the independence of the initial distribution of the concentration, are quite useful. They imply that we can start with a uniformly distributed initial concentration, and that we will get a final result after a finite simulation time. Of course the convergence to the periodic solution is not attained in a final time in an analytic sense, but for a numerical computation we do not have infinite precision anyway, so we can stop the computations after some reasonable time.

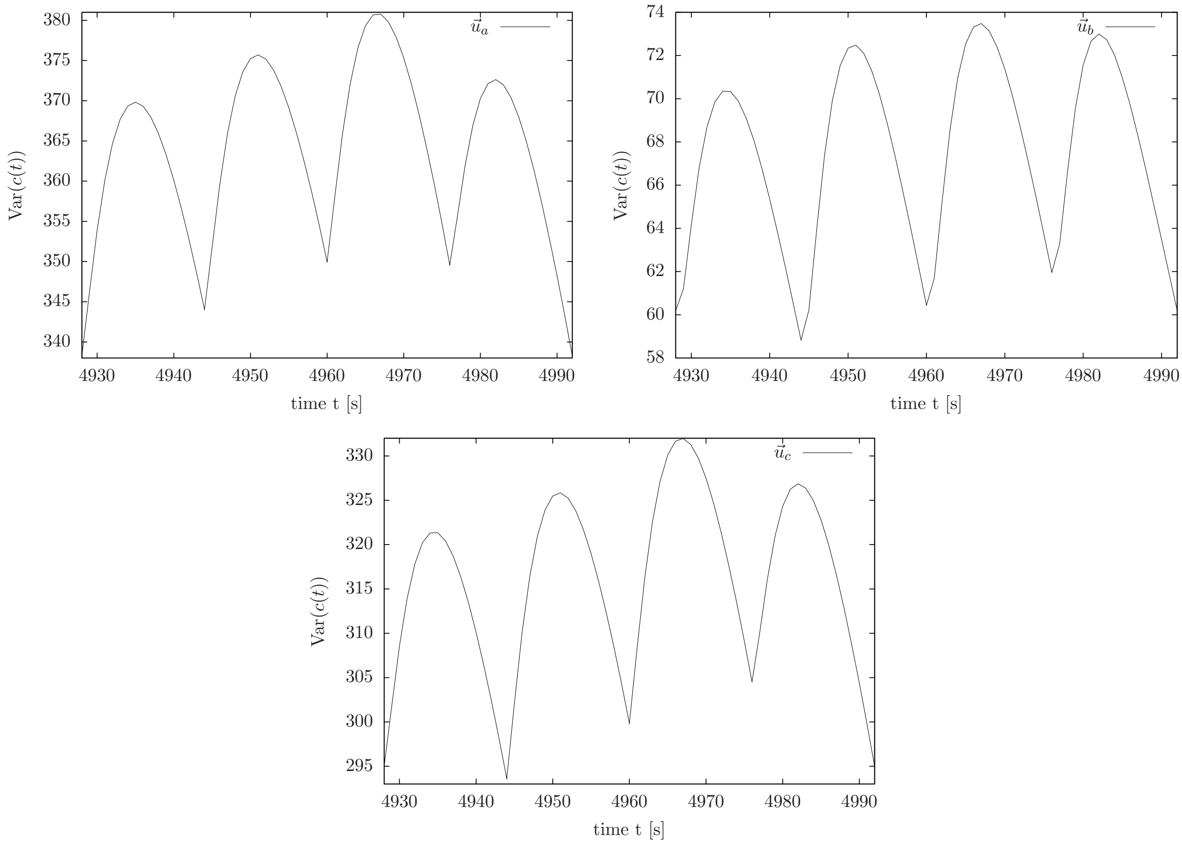
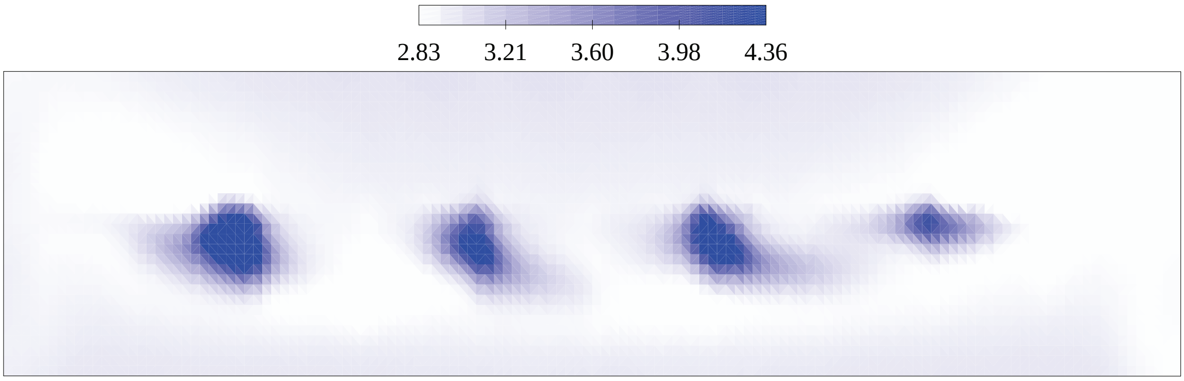
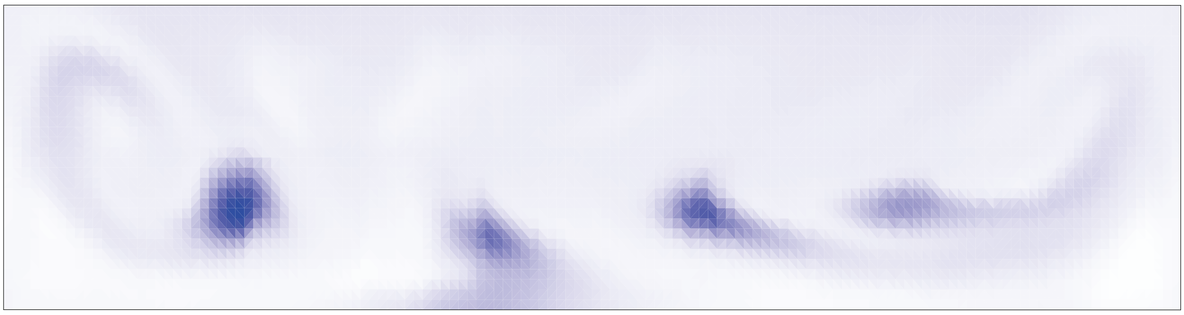


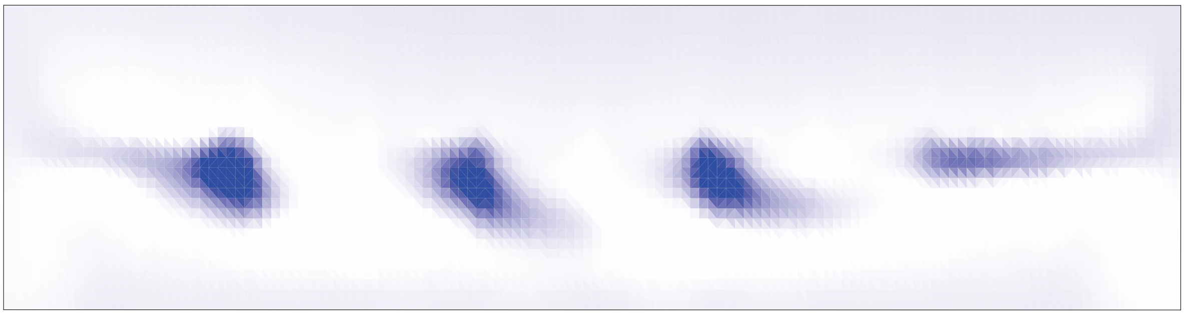
Figure 5.9: The concentration variance in the bath during one feeding cycle for the different velocity fields.



$\vec{u}_a$: $\vec{u} = 0$ on $\partial\Omega$

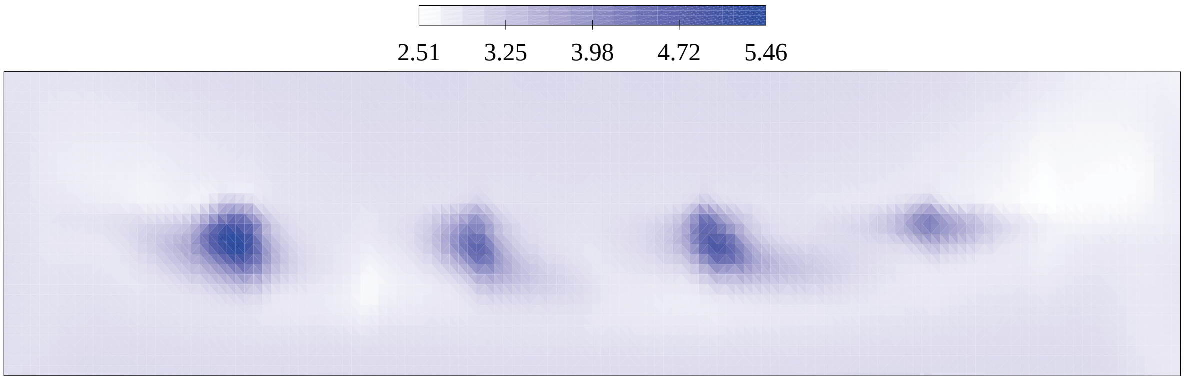


$\vec{u}_b$: $\vec{u} = 0$ on $\partial\Omega$, $\vec{u}.\vec{n} = 0$ at the bottom of the anodes

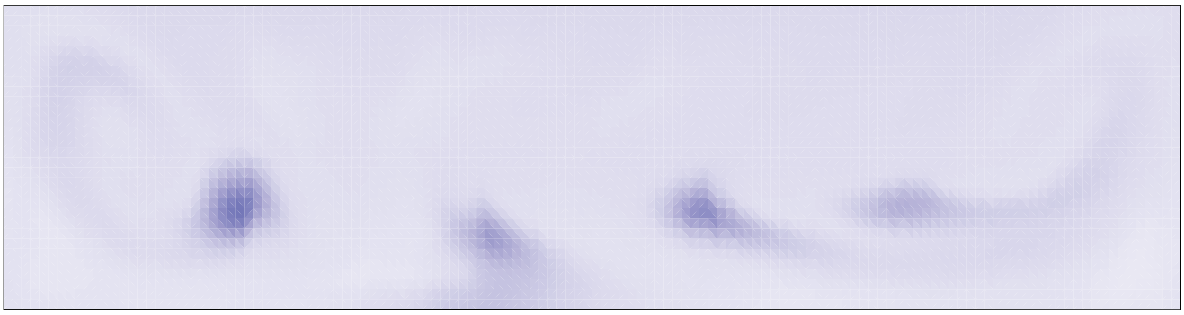


$\vec{u}_c$: $\vec{u} = 0$ on $\partial\Omega$, $\vec{u}.\vec{n} = 0$ at the exterior boundary

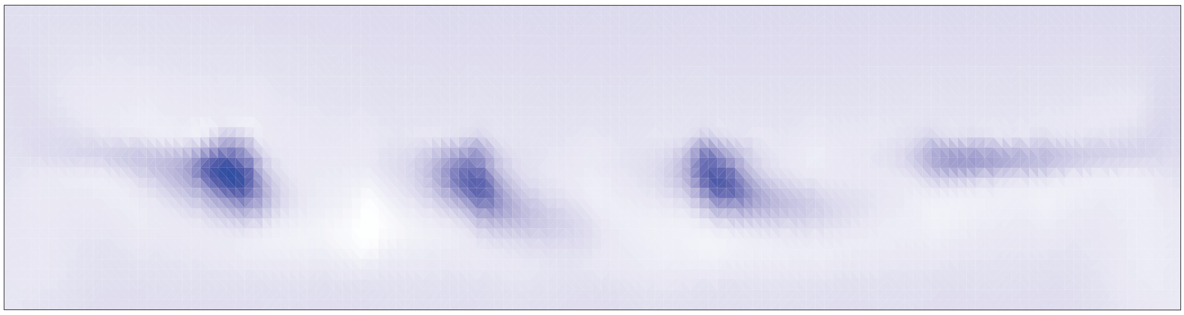
Figure 5.10: Alumina concentration at the periodic state, shown at the bath - aluminium interface, seen from above. Scale with mininum/maximum of the result using $\vec{u}_b$ for all images.



$\vec{u}_a$: $\vec{u} = 0$ on $\partial\Omega$



$\vec{u}_b$: $\vec{u} = 0$ on $\partial\Omega$, $\vec{u}.\vec{n} = 0$ at the bottom of the anodes



$\vec{u}_c$: $\vec{u} = 0$ on $\partial\Omega$, $\vec{u}.\vec{n} = 0$ at the exterior boundary

Figure 5.11: Alumina concentration at the periodic state, shown at the bath - aluminium interface, seen from above. Scale with mininum/maximum of the result using $\vec{u}_a$ for all images.

Chapter 6

Validation of the numerical solution

In order to validate our model and its numerical solution, two experiments on real aluminium electrolysis pots were carried out. We describe shortly the setting of the experiments. Then we compare numerically computed and measured concentrations for both experiments. The chapter ends with a conlusion of the comparison results.

6.1 Description

The experiments were done on a normal working aluminium electrolysis cell. The aim was to measure the total alumina concentration at different points in the bath, over a certain time. During these measurements, the cell was in two different feeding modes.

In the first experiment, the feeding was periodic and such that the alumina concentration should have remained constant over time in the bath. Feeding was done at four different positions, in the order -4.4, 1.6, -1.6, 4.4 [m]. The feeding period was 64 seconds and the four alumina additions took place in each period at times 0, 16, 32, 48 [s], respectively. Each measure was taken when the feeder at -4.4 [m] added its load to the bath. The measures were done at 9 different positions. These positions are depicted by black dots on Figure 6.1, and they are numbered from 1 to 9. Four measures for each position were taken. Each measure was cut into three pieces, and the alumina concentration of every piece was determined. Since the cell was supposed to be in a periodic state, all the results at one position should have been the same. We therefore computed the mean and the standard deviation σ of the 12 results at each position. Those two quantities are reported in columns 2 and 3 of Table 6.1, in the unit weight percents.

In the second experiment we were interested in the alumina distribution when the feeding was turned off, which is called ”control tracking”. The feeding of the cell was stopped, and after 5 minutes 4 measures at the same 9 positions as in the first experiment were taken, the time between two successive measures being one minute. We consider only the last measure, which was thus taken 8 minutes after the feeding was stopped. Since again each measure was cut into three pieces, we compute the mean and the standard deviation σ of the results at each position, which we show in Table 6.2.

For this second experiment we have to adjust our alumina consumption. In the real cell there is a crust of solid cryolite at the boundary of the bath. This crust is not contained in the mesh we use, and therefore the volume of the numerical bath is bigger than the volume of the real bath. The numerical alumina consumption is computed such that the mass of alumina consumed in the numerical problem is equal to the same real quantity. If we denote the total mass of alumina in the bath by m and m_h for the real and the numerical bath, respectively, then the above statement implies that $\dot{m} = \dot{m}_h$ (the dots denote time derivatives). The average alumina concentration in the bath is computed by

$$\bar{c} = \frac{m}{MV}.$$

Since the real volume V is smaller than the numerical volume V_h, we find that

$$\dot{\bar{c}} = \frac{\dot{m}}{MV} > \frac{\dot{m}_h}{MV_h} = \dot{\bar{c}}_h,$$

i.e., the average concentration varies less in the numerical bath than in reality, or here, when we stop the feeding, the average concentration decreases more slowly in the numerical problem than in reality. Since this affects exactly the result we want to verify with the experiment we have to correct the alumina consumption. The weight of the bath is about 7.8 tons in the numerical bath, and about 5.5 tons in the real bath. Hence we multiply the consumption by the factor 7.8/5.5.

The initial concentration of alumina in the bath is not known. When doing the comparisons we start with 3 [wt%] for the first experiment and 2.7 [wt%] for the second experiment. This assumption gives reasonable results, but it could be wrong.

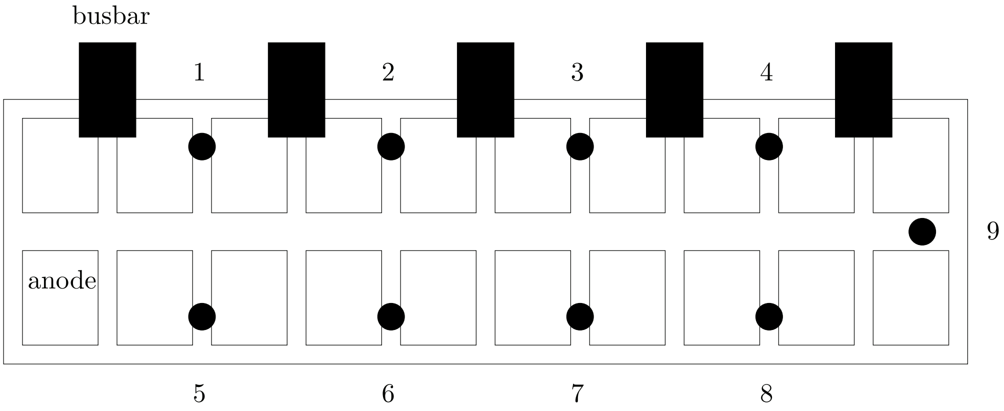


Figure 6.1: A schematic representation of the electrolysis cell, seen from above, and of the positions 1 to 9 at which the measurements of the two experiments were done. The measurement positions are at a distance of about 50 centimetres from the boundary.

6.2 Comparison with the first experiment

When doing the numerical computation we take the setting of the experiment and compute the periodic state. For the first experiment, once this state is attained, we simulate one more feeding cycle and compute the values of the total alumina concentration, i.e., the particle density plus the liquid alumina concentration, at the 9 measurement positions and at the time when the first feeder dumps its load to the bath. Since the results of the experiments are expressed in weight percents, we have to compute the conversion of the concentration c and the particle density n_p to wt%. We first compute the mass of the alumina in a volume V by

$$m_h = M \int_V c_h d\Omega + \frac{4}{3}\pi\rho \sum_{j=1}^{m} R_j^3 \Delta R_j \int_V n_{p,j,h} d\Omega. \tag{6.2.1}$$

position	measure	σ	$\vec{u}_a$	$\vec{u}_b$	$\vec{u}_c$
1	3.11	0.14	2.97	2.97	2.93
2	3.24	0.12	3.01	2.97	2.98
3	3.04	0.12	3.01	2.99	2.98
4	2.92	0.10	2.94	3.01	2.98
5	3.02	0.08	2.93	2.95	2.86
6	2.90	0.06	2.92	3.27	2.86
7	2.91	0.11	2.87	2.92	2.79
8	2.84	0.20	2.95	3.00	2.88
9	2.89	0.08	2.72	3.00	3.09

Table 6.1: Results of the first experiment. We show the measured average alumina concentration in the bath as well as the standard deviation σ of the measured values, followed by the numerical results obtained using different velocity fields. All quantities are in the unit [wt%].

Denoting then the concentration in the unit wt% by c_w, the conversion formula is the following:

$$c_w = \frac{m_h}{m_h\left(1 - \frac{\rho_b}{\rho}\right) + \rho_b V}, \tag{6.2.2}$$

where ρ is the density of alumina and ρ_b the density of the bath. We compute the total alumina concentration in the bath for the three velocity fields $\vec{u}_a, \vec{u}_b$ and $\vec{u}_c$ introduced in Section 5.3. The results are shown in Table 6.1, and, graphically, on Figure 6.2.

Some comments about the results are in order. We note that the behaviour of the results of $\vec{u}_a$ and $\vec{u}_c$ are similar, except at point 9. This can also be seen on Figure 6.3, where we show the total alumina concentration in the bath at the bath - metal pad interface for the two velocity fields. When looking at Figure 6.3 we see the difference at point 9, and a similar difference at the opposite end of the cell, but otherwise the concentration distribution is very similar and also the minimum and maximum values are nearly the same. We notice that the concentration is rather uniformly distributed except for the regions near the feeder positions. The general behaviour of the measured values is quite different from the one computed using $\vec{u}_a$ or $\vec{u}_c$, i.e., the relations between the alumina concentration at different points are often not the same for the measurements and the numerical computations. The largest error for velocity field $\vec{u}_a$ is at position 2, and it is about 7%. For $\vec{u}_c$ this maximal error is about 8% and it is also attained at position 2. In view of the fact that the measurement errors are supposed to be in the order of 10%, these differences between measured and computed values are very satisfying. Thus, the overall behaviour of the numerical solution is different from the measured alumina concentration, but if we compare the values point by point the errors are very small.

The results using velocity field $\vec{u}_b$ are almost equal (except for position 6, see below), and if we check the alumina concentration in the whole bath, shown on Figure 6.4, we see that it is nearly uniformly distributed. We notice that the maximum is lower and the minimum higher than for the two other velocity fields. Again the behaviour of the measure and the numerical solution are different, and again the differences point by point are small. The second largest error, attained at position 2, is about 8% only.

The maximal error is attained at position 6 with 13%. But if we check the alumina distribution at this point on Figure 6.4 we see that position 6 lies in the trace of a freshly added

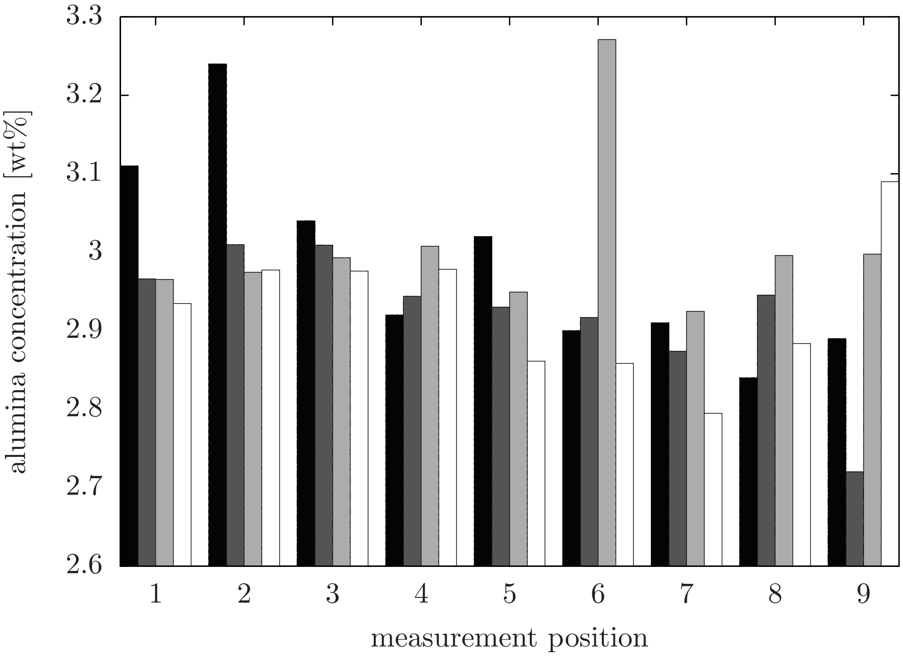


Figure 6.2: Results of the first experiment shown as a bar plot. The measured average alumina concentrations are in black, and the numerical results are in dark grey, light grey and white for the velocity fields $\vec{u}_a$, $\vec{u}_b$ and $\vec{u}_c$ respectively.

particle load. Since this trace will vanish and reappear during each feeding cycle we suppose that the concentration varies strongly over time at this point. To verify this, we plot the time evolution of the concentration at this point over one feeding cycle, shown on Figure 6.5. The value of the concentration in Table 6.1 is taken at time 5008 [s]. We notice that if we had taken the value at 5040 [s], which is half a feeding period later, the alumina concentration would be about 3.16 [wt%]. This shows that the influence of the feeders on the alumina distribution can be important, at least in the numerical model, but probably also in reality. At least the differences between the standard deviations of the measurements are explained by this reason by the engineers who did the experiments.

The measurements were all taken at a distance of about 50 centimetres from the boundary of the bath. Hence, for the numerical results shown in Table 6.1 all the concentrations were computed at 50 centimetres from the boundary. But since the distance between the measurement positions and the boundary of the bath is not exactly 50 centimetres we also compute the alumina concentration at other distances from the boundary. These concentrations, again for position 6 and velocity field $\vec{u}_b$, are plotted on Figure 6.6. We see that the concentration varies strongly during one feeding cycle if the distance to the boundary is between 40 and 60 centimetres, but closer to the boundary the variation is relatively small. We conclude that the distance between the boundary of the bath and the measurement position can have an important impact on the measured concentration. We also note that for most of the measurement points taken here this is not the case in the numerical solution, as can easily be seen on the Figures 6.3, 6.4.

We varied different parameters of the numerical computation in order to get more accurate results. We changed the feeding function in space, adding the load in a smaller or bigger ellipsoid, we used the consumption model including the current density, we varied the diffusivity constant in the range $[4{\cdot}10^{-5},10^{-8}]$, we changed the time for the dissolution of the biggest particles from

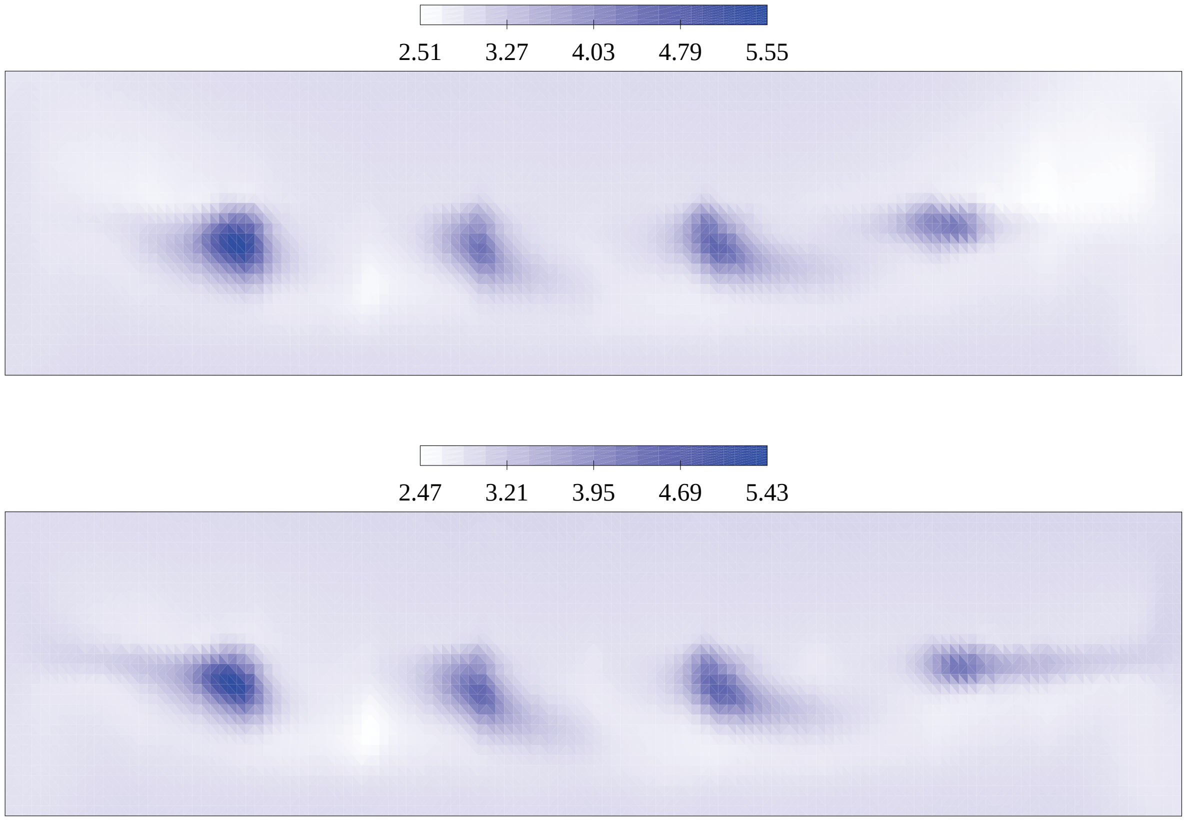


Figure 6.3: Total alumina concentration at the bath - metal pad interface, for the first experiment. Numerical results obtained using $\vec{u}_a$ (above) and $\vec{u}_c$ (below), in their respective linear scales. View from above.

10 to 6 [s], and we used a latency time for the particle dissolution of up to 2 seconds. However, the differences between the various solutions were always very small, in the order of less than 1%. It thus seems that the influence of the velocity field is much stronger than all the other parameters of the model.

We also change the way the velocity field was computed. The fields used here are computed by solving Problem 1.3.4 with the turbulent viscosity model (1.3.5), and we discard the part associated with the bubble functions at the end. If we include the bubble functions in our velocity field, the alumina concentration distribution becomes much less uniform. But especially if we use the laminar viscosity model (1.3.4), the alumina distribution changes a lot compared to the results presented here. The difference between the velocity fields obtained with the turbulent viscosity model and those obtained with the laminar viscosity model is mostly that there are many small vortices when using the turbulent viscosity, which are not present when we use the laminar viscosity. We conclude that the alumina distribution depends very much on the velocity field.

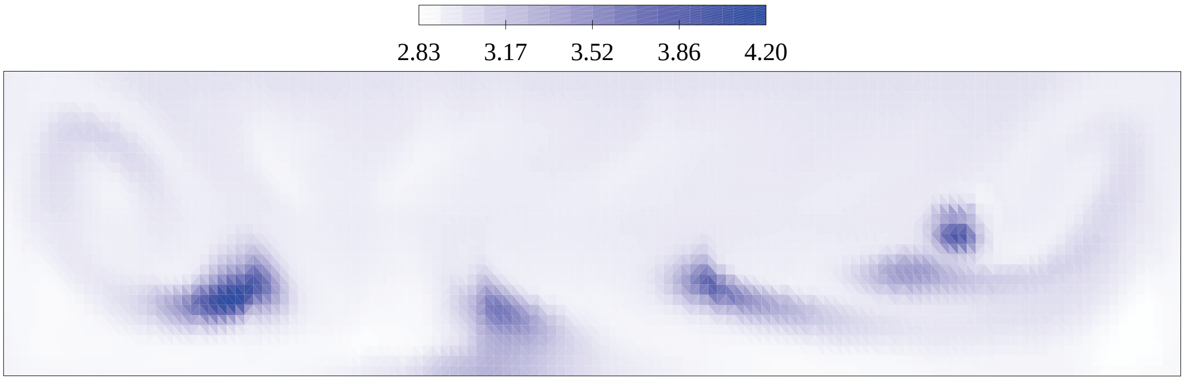


Figure 6.4: Total alumina concentration at the bath - metal pad interface, for the first experiment. Numerical result obtained using $\vec{u}_b$. View from above.

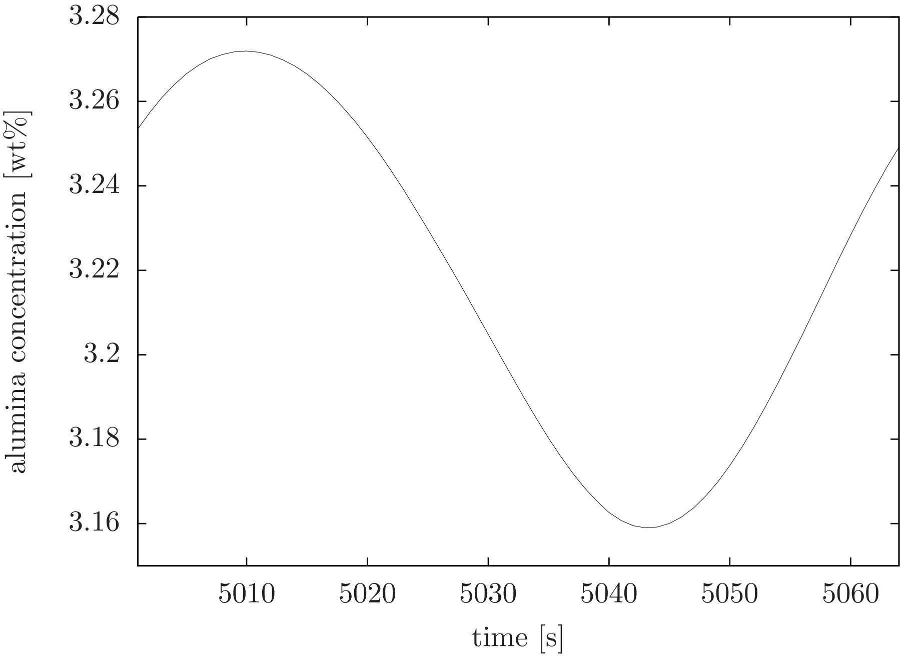


Figure 6.5: The total alumina concentration at position 6, for $\vec{u}_b$.

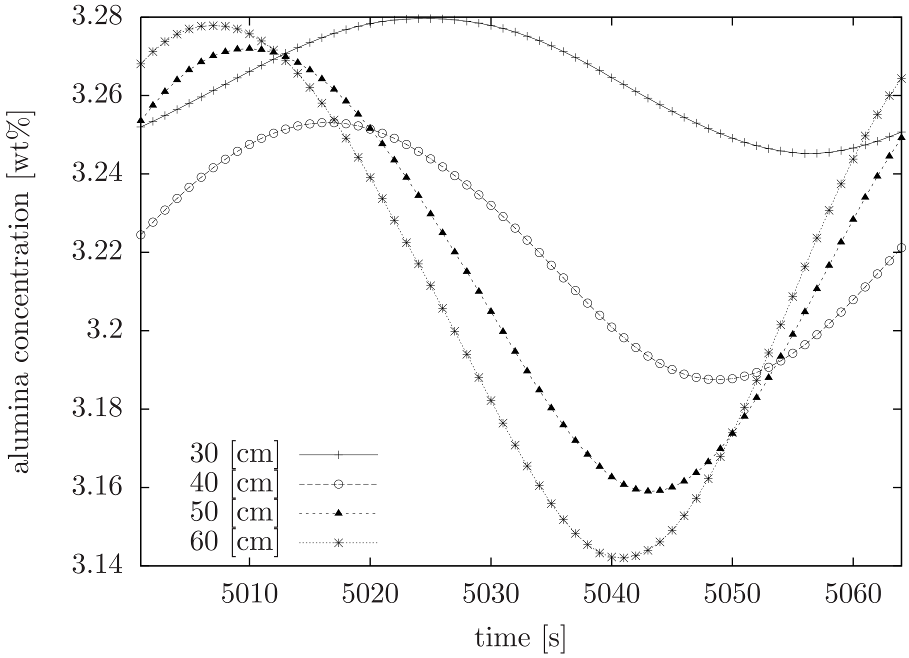


Figure 6.6: The total alumina concentration at position 6, for $\vec{u}_b$. The distance between position 6 and the boundary of the bath is varied from 30 to 60 [cm].

position	measure	σ	$\vec{u}_a$	$\vec{u}_b$	$\vec{u}_c$
1	2.34	0.06	2.33	2.26	2.32
2	2.39	0.04	2.34	2.25	2.33
3	2.21	0.04	2.34	2.26	2.31
4	2.20	0.06	2.29	2.30	2.36
5	2.38	0.03	2.27	2.20	2.23
6	2.34	0.06	2.26	2.11	2.19
7	2.32	0.06	2.23	2.18	2.14
8	2.18	0.01	2.31	2.18	2.27
9	2.08	0.03	2.03	2.17	2.14

Table 6.2: Results of the second experiment. We show the measured average alumina concentration in the bath as well as the standard deviation σ of the measured values, followed by the numerical results obtained using different velocity fields. All quantities are in the unit [wt%].

6.3 Comparison with the second experiment

Now we turn to the second experiment, where we also start from the periodic state, but then stop the alumina feeding. We compute the alumina repartition in the bath 8 minutes after we stopped the feeding and compute the total alumina concentration at the measurement positions. The results obtained for the three velocity fields are in Table 6.2 and, graphically, on Figure 6.7.

The comparison between the measurements and the numerical results of the second experiment lead to a similar conclusion as for the first experiment. The results for velocity fields $\vec{u}_a$ and $\vec{u}_c$ behave quite similarly, see Figure 6.8, but not always in the same way as the measured values. Since the feeding was stopped the concentration is now more uniform than in the first experiment. We notice that the concentration is lowest in the centres of the vortices of the velocity field, by comparing with Figure 5.4. Again, the differences point by point are very small. The maximal error for $\vec{u}_a$ is only 6%, at position 3, and for $\vec{u}_c$ it is 8%, attained at position 7.

Using velocity field $\vec{u}_b$ we still have a very uniform alumina distribution over the whole bath, as shown on Figure 6.9. On the figure there are only two zones where the concentration is much lower than in the rest of the bath, and if we compare with Figure 5.4 we see that these zones are the centres of the large vortices. The smaller vortices seem to have less effect on the alumina concentration than using $\vec{u}_a$ and $\vec{u}_c$. Even if the concentration looks rather uniform the values at the measurement positions vary sensibly, and differently than in the experiment. The maximal error is 10% at position 6, which is still a very good agreement.

Also for this second experiment we changed the parameters of the numerical computation, trying to get results that are closer to the measured values, but again the values at the measurement positions changed less than 0.02 [wt%].

6.4 Conclusion

We used the results of two real world experiments to validate our numerical results and our model. The measurement errors are supposed to be in the order of 10%. In view of this the point by point errors of the numerical solutions, which are in the order of 5 to 10%, are very

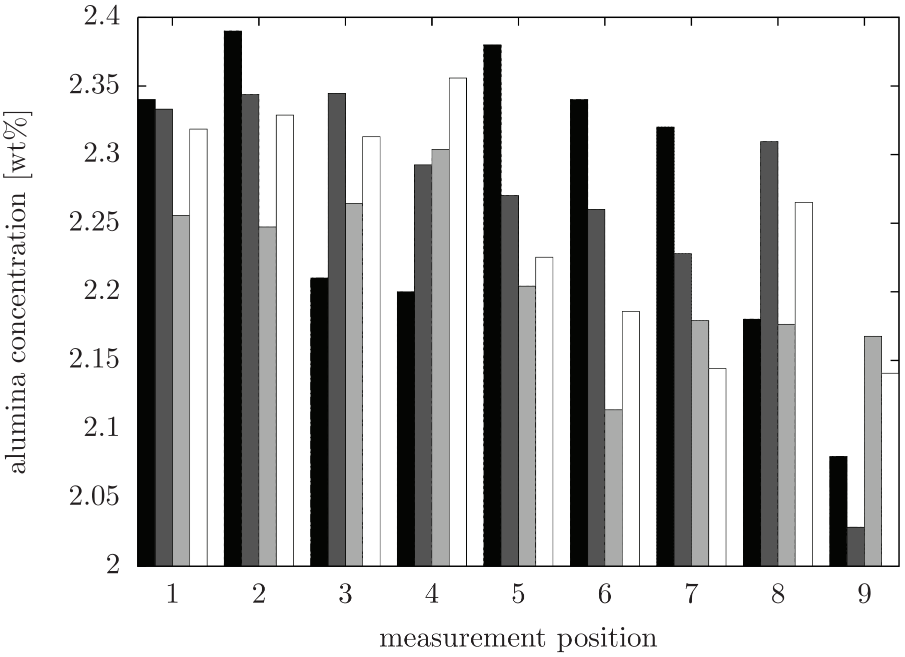


Figure 6.7: Results of the second experiment shown as a bar plot. The measured average alumina concentrations are in black, and the numerical results are in dark grey, light grey and white for the velocity fields $\vec{u}_a$, $\vec{u}_b$ and $\vec{u}_c$ respectively.

good. However, the relative concentration differences between different positions do not match well, i.e., the overall behaviour in the experiment and the numerical solution is rather different. The initial average concentration in the bath is not known for the experiments. We have made assumptions which give well matching results when doing our validation, but we note that the validation would be more significant if the initial average concentration was known.

We notice that the numerical result is sensitive to the distance between the boundary and the measurement position. We also notice that the feeding in the first experiment can have a non-neglectable influence on the pointwise results. This influence is important in the model, but engineers suppose that it is also important when doing the measurements. It is crucial to know the exact consumption when comparing the result of the second experiment with the numerical results.

We find that the numerical result depends mostly on the velocity field. Changing other parameters like the feeding function, the consumption model, the diffusivity coefficient or the particle dissolution speed does not significatively alter the result. Changing the way of how to compute the velocity field can change drastically the alumina concentration distribution.

If further improvements of the model are to be made, more precise measurements are needed, including information about the initial average alumina concentration in the bath. In this case it is also important to validate more precisely the velocity field.

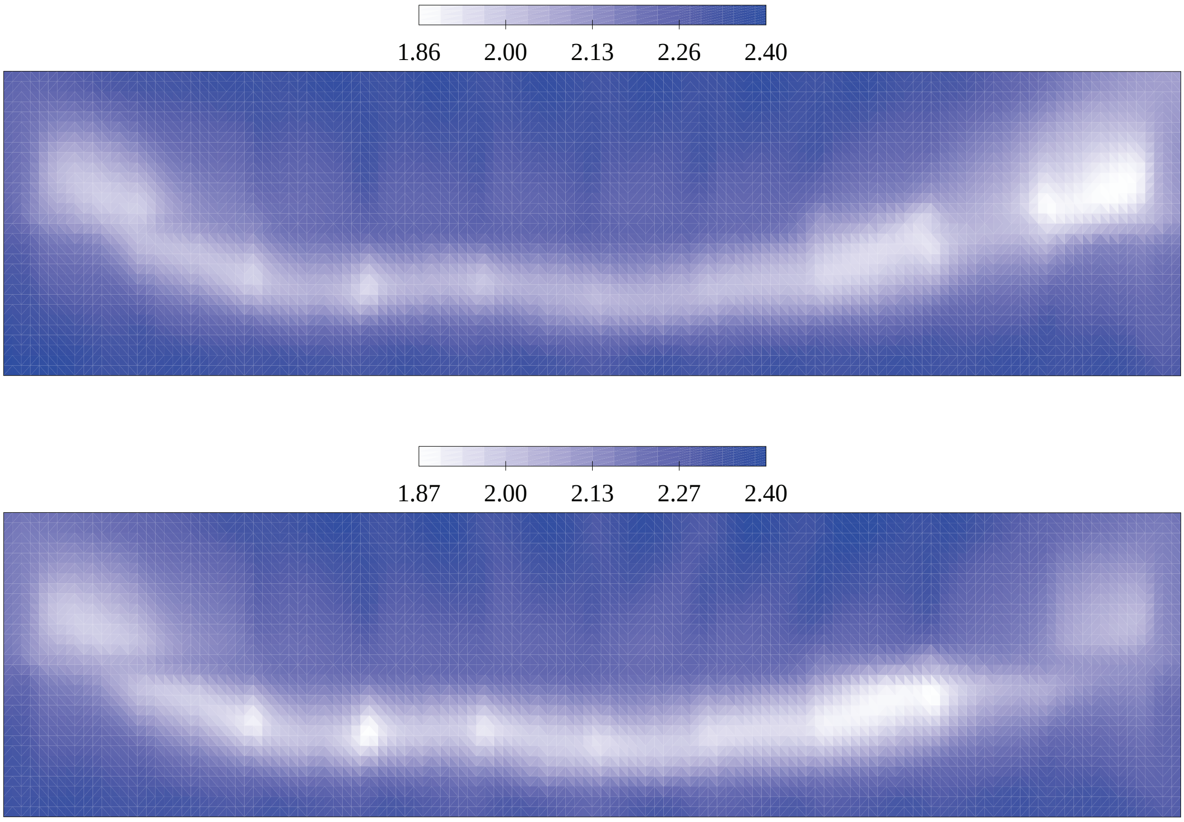


Figure 6.8: Total alumina concentration at the bath - metal pad interface, for the second experiment. Numerical results obtained using $\vec{u}_a$ (above) and $\vec{u}_c$ (below), in their respective linear scales. View from above.

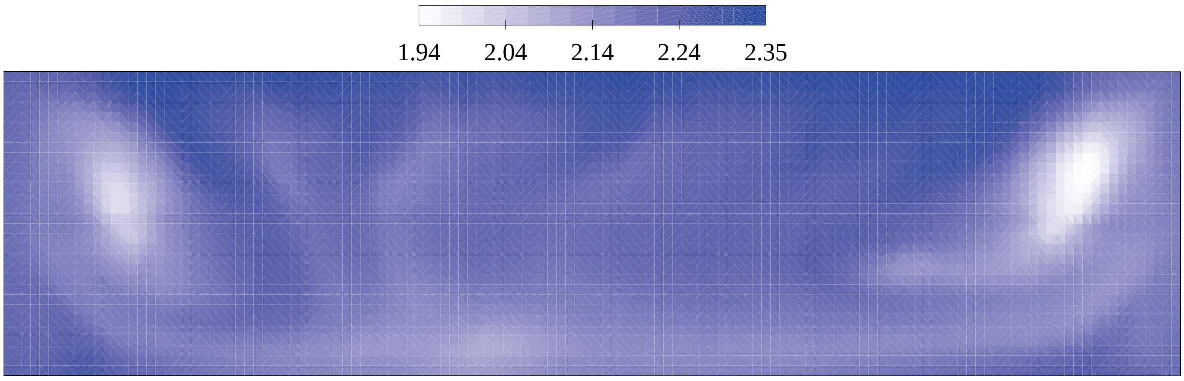


Figure 6.9: Total alumina concentration at the bath - metal pad interface, for the second experiment. Numerical result obtained using $\vec{u}_b$. View from above.

Chapter 7

Conclusion

We developed a model which describes what happens to the alumina in the electrolytic bath of an aluminium production cell. The model starts with the feeding of the alumina, continues with the convection and the dissolution of the particles, treats the convection-diffusion of the emerging liquid alumina concentration and ends with the consumption of the concentration by electrolysis.

The model is discretized using finite elements stabilized by the SUPG method. It is easy to solve the linear systems that appear. However, the underlying velocity field of the bath, computed by solving a $P_1 - P_1$ stabilized discretization of the Navier-Stokes equations, is not divergence free, in such a way that the mass balance of the numerical solution of the alumina problem is not acceptable. One solution of this problem was to stabilize the Navier-Stokes equations by bubble functions, which gave much smaller values for the divergence of the velocity field. Another possibility to ensure a correct mass balance was the use of an original weak formulation of the convection and convection-diffusion problems. This formulation does not only guarantee the correct mass balance, even if the divergence of the velocity field is nonzero, but it also stabilizes the scheme and ensures that the L^2-norm of the solution is bounded by the source term.

With the final discretization, the numerical solution of the alumina problem becomes periodic, and does not depend on the initial condition. This is a very positive result for two reasons. First, we expect the solution to be periodic since this is also the case in reality. And secondly, this allows to optimize a definitive result in the second part, not only some intermediate state, i.e., if we choose the final time of the simulation sufficiently large, the result of the optimization will be independent of this final time and of the initial condition.

Finally, a validation of the numerical results was done by comparing them to results of real world experiments. The conclusion of this comparison was that our numerical results are very close to the measurements, and that they depend mostly on the velocity field.

In summary, we have a simple model of the alumina evolution in the bath, whose discretization is easy to solve and which gives numerical results that are satisfyingly accurate.

If further work was to be done concerning this alumina problem, more precise measurements of real world experiments are needed, and the velocity field has to be validated more precisely. To refine the model, if this was necessary, we would start with the improvement of the feeding of the particles to the bath, and the consumption of liquid alumina by electrolysis. The model of these two parts are rather coarse so far. Then, one could think of the introduction of buoyancy forces acting on the particles, or consider the inertia and the clustering of the particles.

Part II

Optimization

Chapter 8

Problem formulation

As pointed out in the introduction, we would like the alumina concentration in the bath to be as uniform as possible. In this part, we proceed to reach this aim by optimization. We will first describe the physical problem. In Section 8.2, we will give a very short introduction to optimization theory. Then we will formulate our problem mathematically. At the end, several specific optimization problems will be presented.

8.1 Physical problem

In the initial description of the aluminium reduction process, we noted the importance of having the alumina concentration in the bath constantly between 1.5 and 3 [wt%]. If the concentration is above this interval, the solid alumina added to the bath will not dissolve and sink to the bottom of the cell, with the result that the current efficiency decreases. In the case of too low concentration, the cell goes into anode effect, meaning that the aluminium production will be interrupted and green house gases will be produced instead.

A rather uniform alumina concentration in the bath is therefore important for the efficiency of the aluminium production. Based on the simulation of the concentration presented in the first part, we will now vary different parameters of the alumina feeding system in order to optimize the uniformity of the concentration in the bath. Our aim is a constant concentration over the whole bath, varying only periodically in time according to the feeding period. This constant concentration should be attained when the cell is in its periodic state.

There are several parameters which define the feeding and which can be changed. We suppose that the feeding is done periodically, with each period having the same length and having the same parameters. For example, if we have four feeders in a cell, we can fix the length of one feeding period, in which each feeder will add one alumina load to the bath. We also have to fix the position of each feeder, the time, when each feeder gives its alumina load to the bath, and the size of each alumina load. But we can only fix the parameters of the first feeding period, and all the following periods will be exactly the same as the first one. The parameters to vary are thus the feeder positions, the feeding times, the load sizes and the feeding period length.

We will again make the simplifying assumption that the alumina concentration does not have any influence on the cell. In particular, we suppose that the current efficieny remains constant and that the alumina consumption by electrolysis is uniform over the part of the bath lying under the anodes. The only exception to this last statement occurs when the local concentration goes below zero, in which case the consumption in the domain with negative concentration is zero and the total consumption is uniformly distributed on the rest of the bath below the anodes, i.e., we use the threshold consumption model of Section 4.2.

8.2 Short introduction to optimization

The mathematical formulation of an unconstrained optimization problem is given by

$$\min_{\vec{x}} J(\vec{x}), \tag{8.2.1}$$

where $\vec{x} \in \mathbb{R}^n$ is the vector of optimization variables and J is the objective function.

We are trying to find a global minimizer of J. We have the following

Definition 8.2.1. A point $\vec{x}^*$ is a global minimizer if

$$J(\vec{x}^*) \leq J(\vec{x}), \quad \forall \vec{x} \in \mathbb{R}^n. \tag{8.2.2}$$

Since we do not have information about the overall shape of J we usually cannot assure that the solution found is a global minimum. Most algorithms are able to find only local minima.

Definition 8.2.2. A point $\vec{x}^*$ is a local minimizer if there exists a neighbourhood N of $\vec{x}^*$ such that

$$J(\vec{x}^*) \leq J(\vec{x}), \quad \forall \vec{x} \in N. \tag{8.2.3}$$

Analogically, we have a strict local minimizer defined by

Definition 8.2.3. A point $\vec{x}^*$ is a strict local minimizer if there exists a neighbourhood N of $\vec{x}^*$ such that

$$J(\vec{x}^*) < J(\vec{x}), \quad \forall \vec{x} \in N \text{ with } \vec{x} \neq \vec{x}^*. \tag{8.2.4}$$

Following these definitions, one would have to check all possible $\vec{x}$ in a neighbourhood to assure a local minimum. When the function J is smooth, there are more efficient ways to identify local minima. We are most interested by sufficient conditions which guarantee that $\vec{x}^*$ is a local minimizer. One such condition is given by the following

Theorem 8.2.4. *Suppose that $\nabla^2 J$ is continuous in an open neighbourhood of $\vec{x}^*$ and that $\vec{\nabla} J(\vec{x}^*) = 0$ and $\nabla^2 J(\vec{x}^*)$ is positive definite. Then $\vec{x}^*$ is a strict local minimizer of J.*

Here, $\nabla^2 J$ denotes the Hessian matrix of J. For a proof of the theorem, see for example [29, theorem 2.4].

8.3 Mathematical formulation of the optimization problem

The aim is a uniform concentration in the bath, varying only periodically in time according to the feeding period. It will certainly not be possible to reach a uniform concentration, because of the accumulations created by freshly added particles that are dissolving, but we want to have as little variations as possible in the concentration field. The idea is to define an objective function J which measures the distance of the concentration distribution to a uniform concentration. The objective function will be positive, and when it is zero the concentration is uniform in the bath. We then want to minimize J.

There are several ways to compute the variation of the concentration. In analogy with the computation of statistical variance we propose

$$J = \|c - \bar{c}\|_a, \tag{8.3.1}$$

with $\|.\|_a$ a norm that we will specify below. The objective function J will be considered as a function of the optimization variables, which in any case will not appear explicitly in the concentration. We denote by $\bar{c}(t)$ the mean of the concentration over the bath at time t, i.e.,

$$\bar{c}(t) = \frac{1}{|\Omega_{\mathrm{el}}|} \int_{\Omega_{\mathrm{el}}} c(\vec{x}, t) d\Omega. \tag{8.3.2}$$

The choice of the time t at which to measure J is arbitrary if we are at the periodic state of the cell, when c is periodic. The objective function J will then forcibly be a periodic function. In order to control the variance of c over one complete feeding period, we will take the mean of $\|c(t) - \bar{c}(t)\|_b$ over one period, where $\|.\|_b$ is some norm in space. Thus, denoting the feeding period by ΔT, we define the norm $\|.\|_a$ by

$$\|c\|_a = \frac{1}{\Delta T} \int_{T-\Delta T}^{T} \|c(t)\|_b dt. \tag{8.3.3}$$

Here, T is the final time of the simulation. We also have to define the norm in space. We would like to have a norm which gives a "nice" objective function when varying the optimization parameters. Here, "nice" means smooth and with clear differences between minima and maxima, and if possible there should be a small number of local minima in order to simplify the minimization. All of the L^p norms seem to be adequate candidates and we choose the mean of the L^2 norm. Different L^p norms will be used for comparison in Section 10.1 and we will justify our choice. Hence, our objective function will in general be defined by

$$J = \frac{1}{\Delta T |\Omega_{\mathrm{el}}|} \int_{T-\Delta T}^{T} \int_{\Omega_{\mathrm{el}}} (c_h(\vec{x})(t) - \bar{c}_h(t))^2 d\Omega dt. \tag{8.3.4}$$

We showed in the first part in Section 5.3 that a periodic solution of the concentration is reached. But starting from a uniform concentration, the time to reach this periodic state might be rather long. For example, when working with velocity field $\vec{u}_a$ or $\vec{u}_c$, defined in Section 5.3, we saw that it takes about 4000 simulated seconds to reach the periodic state, and the computation time for this simulation is about two hours. Computing the value of the objective function J means simulating the alumina feeding up to the periodic state. At this point J will be constant, since we take the mean over the feeding period. For an optimization, where we want to compute the objective function many times for different values of the optimization variables, this final time is large, the optimization would take a very long time. When looking at the simulation, we see that the concentration reaches an almost periodic state long before it remains strictly periodic, meaning that the objective function will not change much from a certain point on. We will therefore work with a final time T which is only 10 times the length of the feeding period. A debate about the validity of this assumption will be held in the results section.

We are not so much interested in the minimal value of the objective function that we can reach, but more in the values of the optimization variables which give that minimal value. These results should be independent of the discretization of the simulation. This implies that we have to use a fine discretization when we optimize. At the same time, as noted above, we will have to compute the simulation many times and thus we want to have a coarse discretization which gives fast results.

In the following we will give specific optimization problems with different optimization variables. When minimizing the objective function with respect to certain variables, for example the feeder positions, then all the other variables of the feeder configuration, for example the load weights and the feeding times, are fixed. We could also minimize J with respect to all variables of the feeder configuration, but it would take a very long time to do this, because the number of variables is much larger, and thus the complexity of the optimization problem is much higher.

Optimization of the feeder positions

First, we will consider an optimization with respect to the position of the alumina feeders. We suppose that the feeders can take any position in the central channel. Thus, for any feeder we can define the x-component of its position. We set the origin of the x-axis in the middle of the cell and define $2L$ to be the length of the cell. Denoting the optimization variables by $x_1, \dots, x_n$ we have the following optimization problem:

$$\min_{(x_1,\dots,x_n)} J(x_1, \dots, x_n) \tag{8.3.5}$$

$$-L \le x_i \le L, \quad 1 \le i \le n. \tag{8.3.6}$$

Since we prefer to have an unconstrained problem, we introduce the following change of variables:

$$x_i' = \tan\left(\frac{\pi}{2}\frac{x_i}{L}\right), \quad 1 \le i \le n. \tag{8.3.7}$$

Hence, $-\infty < x_i' < \infty$, and the optimization problem becomes simply

$$\min_{(x_1',\dots,x_n')} J'(x_1', \dots, x_n'). \tag{8.3.8}$$

We note here that the optimum to be determined is not found for $x_i = \pm L$, i.e., for $x_i' = \pm\infty$, and thus the optimum can be determined in the modified variables.

Optimization of the feeding times

Here, we want to optimize the objective function with respect to the alumina feeding times. The feeding period is fixed to ΔT. One feeding time is arbitrary since we have a feeding cycle. All other feeding times are between 0 and ΔT. Denoting the optimization variables by $\tau^1, \dots, \tau^{n-1}$ we have the following optimization problem:

$$\min_{(\tau^1,\dots,\tau^{n-1})} J(\tau^1, \dots, \tau^n) \tag{8.3.9}$$

$$0 \le \tau^i \le \Delta T, \quad 1 \le i \le n-1. \tag{8.3.10}$$

Again we can make a change of variables

$$\tau^{i\prime} = \tan\left(\pi\frac{\tau^i}{\Delta T} - \frac{\pi}{2}\right), \quad 1 \le i \le n-1, \tag{8.3.11}$$

to get an unconstrained optimization problem

$$\min_{(\tau^{1\prime},\dots,\tau^{n-1\prime})} J''(\tau^{1\prime}, \dots, \tau^{n\prime}). \tag{8.3.12}$$

However, when we computed some values of J for different feeding times we noticed that the difference of these values is very small, i.e., in the order of 1-2%. We did some further tests on this problem. Our conclusion regarding this optimization problem is twofold. First, if we want to solve this problem we have to work with integer values for the optimization variables τ^i, because otherwise the objective function shows a singularity when τ^i passes from noninteger to integer values. The function is continuous, but it is not differentiable, and the optimization of this function, which has a peak at every integer value of τ^i, is very difficult. And secondly, more important, we noticed that if the added total weight of alumina was correct, the influence of the feeding times on J was very small, no matter which feeding times were chosen. Since it does not make sense to minimize a function where the minimum value is more than 95% of the maximum value, or, put differently, where the maximum gain is less than 5%, we will not pursue this problem.

Optimization of the load weights

Another parameter we can change is the alumina load weight of each feeder. Since we suppose that the current efficiency is constant, the total mass of alumina consumed is known. Thus we have to make sure that the sum of the alumina loads of all feeders corresponds to the mass consumed during one feeding period, denoted by M. We denote the alumina loads by $w_1, \ldots, w_n$. The optimization problems becomes the following:

$$\min_{(w_1,\ldots,w_n)} J(w_1,\ldots,w_n) \tag{8.3.13}$$

$$0 \leq w_i \leq M, \quad 1 \leq i \leq n, \tag{8.3.14}$$

$$\sum_{i=1}^{n} w_i = M. \tag{8.3.15}$$

So here we not only have bounds on each variable but also a constraint on the sum of the variables. We introduce a similar change of variables as before

$$w_i' = \tan\left(\pi \frac{w_i}{M} - \frac{\pi}{2}\right), \quad 1 \leq i \leq n-1, \tag{8.3.16}$$

which gives us $-\infty < w_i' < \infty, i = 1, \ldots, n-1$. The problem here is that we do not control the sum of the weights. If we set $w_n = M - \sum_{i=1}^{n-1} w_i$ we are sure that $\sum_{i=1}^{n} w_i = M$, but it is possible that $w_n < 0$, which is not realistic. And if we impose

$$w_n = \max(M - \sum_{i=1}^{n-1} w_i, 0) \tag{8.3.17}$$

then we ensure $\sum_{i=1}^{n} w_i \geq M$, but the inequality can be strict, hence, it is possible that we add too much alumina. However, when doing computations with setting w_n as in (8.3.17), the value of J increases when the sum of the weights goes beyond M. Thus, when minimizing J with respect to the load weights, the minimum is always found for weights which respect $\sum_{i=1}^{n} w_i = M$. This is not a complete surprise, because if we add too much alumina during each feeding cycle, the total amount of alumina in the bath will increase. Hence, the differences between regions with a high or low alumina concentration will become more pronounced, and the variance J will thus be higher. In what follows we will therefore consider the problem

$$\min_{(w_1',\ldots,w_{n-1}')} J^*(w_1',\ldots,w_n') \tag{8.3.18}$$

$$w_n' = \tan\left(\max\left(\frac{1-n}{2}\pi - \sum_{i=1}^{n-1} \arctan(w_i'), 0\right) - \frac{\pi}{2}\right), \tag{8.3.19}$$

where (8.3.19) is equivalent to (8.3.17), but is here written in the new variables.
If the objective function would not admit such a simplification, we would have to work with a constrained optimization problem, which could be solved using Lagrange multipliers.

Optimization of the feeding period

Finally, we can also change the feeding period. We have to fix a minimum feeding period ΔT_m because the feeders can not add alumina continuously. And we also fix a maximum period ΔT_M. By making the change of variables

$$\Delta T' = \tan\left(\pi \frac{\Delta T - \Delta T_m}{\Delta T_M - \Delta T_m} - \frac{\pi}{2}\right) \tag{8.3.20}$$

we get the unconstrained optimization problem

$$\min_{\Delta T'} J^{**}(\Delta T'). \tag{8.3.21}$$

We note that this is an optimization problem with one single variable. Doing some tests we notice that the minimum of J is always attained for the minimum feeding period ΔT_m. This is not so much surprising, because we think that the alumina particles are best distributed if they are continuously added to the bath, which is not possible for technical reasons. Thus, we will skip this problem because the solution is always the shortest technically possible period.

Chapter 9

Optimization algorithms

We will start this chapter by explaining the basics of line search methods and in particular of Newton's method. In the following sections, we will give a short overview of automatic differentiation and continue by describing the Particle Swarm Optimization (PSO) algorithm. In the last section, we will discuss some of the fundamental differences between Newton's method and PSO.

9.1 Newton's method and line search

Line search methods are iterative methods for solving optimization problems. At each step, one chooses a direction $\vec{p}_k$, and, from the current iterate $\vec{x}_k$, tries to find a step length α which solves

$$\min_{\alpha>0} J(\vec{x}_k + \alpha\vec{p}_k). \tag{9.1.1}$$

The next iterate is then $\vec{x}_{k+1} = \vec{x}_k + \alpha\vec{p}_k$. This one-dimensional minimization problem is not solved exactly because this might be too expensive and it is usually not necessary. Instead, a step length is computed by the line search algorithm which approximates the minimum. At the new iterate a new search direction and a new step length are computed, and this process is repeated.

Since we do not want to solve the line search problem exactly, we need a criterion that tells us whether a certain step length approximates sufficiently well the minimizer or not. We face a tradeoff because we want to reduce J as much as possible but in as few iterations as possible. One possibility is the following. Let $\phi(\alpha) = J(\vec{x}_k + \alpha\vec{p}_k), \alpha \geq 0$. If $\phi'(0) < 0$, given $\mu, \eta \in (0,1)$ we want to find $\alpha > 0$ such that

$$\phi(\alpha) \leq \phi(0) + \alpha\mu\phi'(0), \tag{9.1.2}$$

$$|\phi'(\alpha)| \leq \eta|\phi'(0)|. \tag{9.1.3}$$

The first condition forces a sufficient decrease in the function, see Figure 9.1 for an illustration. However, this condition allows arbitrary small choices of $\alpha > 0$. The second condition rules out small choices of α. It is a curvature condition because it implies that $\phi'(\alpha) - \phi'(0) \geq (1-\eta)|\phi'(0)|$, and thus, that the average curvature of ϕ on $(0, \alpha)$ is positive. An illustration of this condition is shown on Figure 9.2. Moré and Thuente [28] describe an algorithm which produces a sequence of iterates that converge to an α which satisfies conditions (9.1.2) and (9.1.3) when $\mu \leq \eta$.

In Newton's method, the search direction is given by

$$\vec{p}_k = -(\nabla^2 J(\vec{x}_k))^{-1}\vec{\nabla} J(\vec{x}_k). \tag{9.1.4}$$

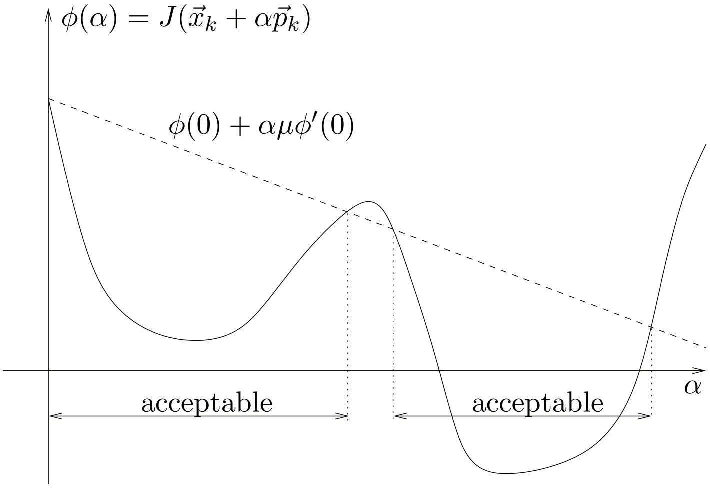

$\phi(\alpha) = J(\vec{x}_k + \alpha\vec{p}_k)$
$\phi(0) + \alpha\mu\phi'(0)$
α
acceptable
acceptable

Figure 9.1: Sufficient decrease condition

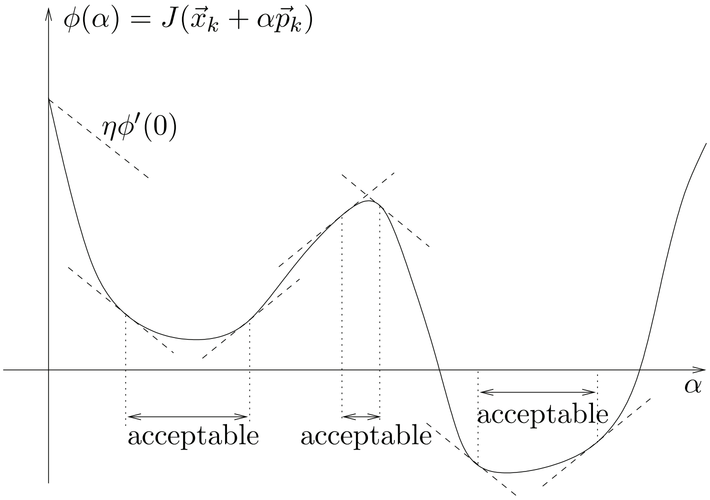


Figure 9.2: Sufficient curvature condition

We know that for all $\vec{x}$ in a neighbourhood of a minimum $\vec{x}^*$ such that $\nabla^2 J(\vec{x}^*)$ is positive definite, the Hessian $\nabla^2 J(\vec{x})$ will also be positive definite. Newton's method will be well defined in this region. We have the following

Theorem 9.1.1. *Suppose that J is twice differentiable and that $\nabla^2 J(\vec{x})$ is Lipschitz continuous in a neighbourhood of a solution $\vec{x}^*$ at which the sufficient conditions of Theorem 8.2.4 are satisfied. Consider the iteration $\vec{x}_{k+1} = \vec{x}_k + \vec{p}_k$, where $\vec{p}_k$ is defined by* (9.1.4)*. Then*

1. *if the starting point $\vec{x}_0$ is sufficiently close to $\vec{x}^*$, the sequence of iterates converges to $\vec{x}^*$;*

2. *the rate of convergence of $\{x_k\}$ is quadratic.*

For a proof, see for example [29, theorem 3.5].

For our computer implementation we use the library TAO (Toolkit for Advanced Optimization) [5], which is part ot the PETSc project developed by Argonne National Laboratory [4]. TAO implements the line search algorithm with the stopping criteria described in [28]. The algorithm needs to evaluate the function, gradient and Hessian of the objective function for certain values of the optimization variables, and using this information it computes new directions using Newton's algorithm. We compute the gradient and Hessian of J using automatic differentiation. A short introduction to this method is given in the next section.

9.2 Automatic differentiation

We want to differentiate the function J which depends on several variables, for example on the positions of the feeders x_i. The values of J are computed with a computer program which takes x_i as input variables and gives $J(x_i)$ as the output. The program might do a lot of steps during the computation, but every step is a very elementary operation, for example a multiplication, or computing the sinus of an argument. To differentiate J with respect to x_i we can apply the chain rule to the computer program, i.e., every elementary operation will be differentiated separately and multiplied by the result of the previous step. Only numerical values are stored, we do not compute a symbolic derivative. The differentiation will take a lot of steps, but every step is easy to compute, like in the underlying computer program. This way of obtaining the derivative is called automatic differentiation. According to [19], automatic differentiation started already with Newton and Leibniz, and has since then been rediscovered several times.

There are two ways of computing derivatives: the forward mode and the reverse mode. In the next two subsections we explain by examples how the forward and the reverse modes work. For more details about automatic differentiation the reader is referred to the book written by Griewank and Walther [19].

At the end of this section, we discuss the choice of the differentiation mode for our optimization problem.

Forward mode by example

We want to differentiate $y = f(x) = \sin(x^2)$ for $x = 2$. y could be computed in the following way:

$$\begin{aligned} v_0 &= x = 2 \\ v_1 &= v_0^2 = 4 \\ y = v_2 &= \sin(v_1) = -0.75680. \end{aligned}$$

We introduce for each variable v_i the variable $\dot{v}_i = \frac{\partial v_i}{\partial x}$. $\dot{x}$ is initialized to 1 since we want to differentiate f with respect to x. We compute

$$\begin{aligned}
\dot{v}_0 &= \dot{x} = 1 \\
\dot{v}_1 &= 2v_0\dot{v}_0 = 4 \\
\dot{v}_2 &= \cos(v_1)\dot{v}_1 = -2.6146.
\end{aligned}$$

Thus, we find $f'(x)|_{x=2} = -2.6146$. Every step of the differentiation could be done just after the corresponding step of the function evaluation, i.e., $\dot{v}_i$ can be computed just after v_i has been calculated. Therefore, computing the derivative of f with respect to x can be done at the same time as evaluating $f(x)$. The time to compute the derivative is proportional to the time to evaluate the function.

If the function $\vec{f}$ is a vector-valued function, the forward mode computes the derivative of each component in one sweep, i.e., the time to compute the derivative does not depend on the number of components of $\vec{f}$.

On the other hand, if f depends on a vector $\vec{x}$ we have to repeat the derivative step for every component of $\vec{x}$. This implies that the time to compute the derivative is multiplied by the number of components of $\vec{x}$.

As an illustration for the two last assertions we consider the vector-valued function

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \vec{F}(x_1, x_2) = \begin{pmatrix} x_2 \sin(x_1^2) \\ e^{x_1 x_2} \end{pmatrix}, \tag{9.2.1}$$

which we want to evaluate at $(x_1, x_2) = (2, 3)$. We set

$$\begin{aligned}
v_0 &= x_1 = 2 \\
v_1 &= x_2 = 3
\end{aligned}$$

and we compute

$$\begin{aligned}
v_2 &= v_0^2 = 4 \\
v_3 &= \sin(v_2) = -0.75680 \\
v_4 &= v_0 v_1 = 6 \\
y_1 = v_5 &= v_1 v_3 = -2.2704 \\
y_2 = v_6 &= e^{v_4} = 403.43.
\end{aligned}$$

To compute the derivative with respect to x_1 we set

$$\begin{aligned}
\dot{v}_0 &= \dot{x}_1 = 1 \\
\dot{v}_1 &= \dot{x}_2 = 0
\end{aligned}$$

and we compute

$$\begin{aligned}
\dot{v}_2 &= 2v_0\dot{v}_0 = 4 \\
\dot{v}_3 &= \cos(v_2)\dot{v}_2 = -2.6146 \\
\dot{v}_4 &= \dot{v}_0 v_1 + v_0 \dot{v}_1 = 3 \\
\dot{v}_5 &= \dot{v}_1 v_3 + v_1 \dot{v}_3 = -7.8437 \\
\dot{v}_6 &= \dot{v}_4 e^{v_4} = 1210.3.
\end{aligned}$$

We get the partial derivative of both components of $\vec{F} = (F_1, F_2)$ with respect to x_1,

$$\frac{\partial F_1}{\partial x_1}(2,3) = -7.8437, \qquad \frac{\partial F_2}{\partial x_1}(2,3) = 1210.3.$$

The cost of the computation of these two derivatives is equal to the cost of the function evaluation.

If we wanted to compute the derivatives with respect to x_2 we would have to do analogous computations as for x_1 again, multiplying thus the computational effort for the differentiation by 2.

Reverse mode by example

Again we want to differentiate the function $y = f(x) = \sin(x^2)$ introduced in the previous section. We recall the way y is computed for $x = 2$:

$$\begin{aligned} v_0 &= x = 2 \\ v_1 &= v_0^2 = 4 \\ y = v_2 &= \sin(v_1) = -0.75680. \end{aligned}$$

This time, we introduce to each variable v_i the variable $\bar{v}_i = \frac{\partial y}{\partial v_i}$. We initialize $\bar{y}$ to 1 since we want to differentiate y with respect to x. We compute

$$\begin{aligned} \bar{v}_2 &= \bar{y} = 1 \\ \bar{v}_1 &= \cos(v_1)\bar{v}_2 = -0.65364 \\ \bar{v}_0 &= 2v_0\bar{v}_1 = -2.6146. \end{aligned}$$

We find the same value for $f'(x)|_{x=2}$, but this time we had to know all the intermediate computations for f to be able to differentiate. This implies that the function has first to be evaluated and only then it is possible to compute the derivative using reverse mode.

There are two main possibilities to compute the derivative in reverse mode: either we first compute f and store all intermediate values, or we compute the intermediate values each time we need them. The first possibility is illustrated on Figure 9.3. We compute $v_1, v_2, \ldots, v_n$ and we store all these values. Then we compute $\bar{v}_n, \bar{v}_{n-1}, \ldots, \bar{v}_0$. The time to compute the derivative is proportional to the time to evaluate the function, but since all intermediate values of the function have to be stored, the memory requirement could be very high. The second possibility is illustrated on Figure 9.4. Here we compute v_n, $\bar{v}_n$, v_{n-1}, $\bar{v}_{n-1}$, and so on, until $\bar{v}_0$. The time to compute the derivative is proportional to the square of the time to evaluate the function, but very little memory is used.

Hence, we either store all intermediate values and use a lot of memory, or we compute all intermediate values when needed and use a lot of time. The choice depends on the computation which needs to be done. It is also possible to combine the two possibilities, i.e., to store only a part of the intermediate results and to recompute the other values when needed.

If the function $\vec{f}$ is a vector-valued function, the reverse mode has to be computed once for every component of $\vec{f}$, i.e., the time to compute the derivative is proportional to the number of components of $\vec{f}$.

On the other hand, if f depends on a vector $\vec{x}$, the derivative is computed in one sweep, implying that the time to compute the derivative is independent of the number of components of $\vec{x}$.

Again to illustrate the two last assertions we consider the function defined in (9.2.1). The function evaluation is still the same. To compute the derivative of F_1 we set

$$\begin{aligned}\bar{v}_6 &= \bar{y}_2 = 0\\ \bar{v}_5 &= \bar{y}_1 = 1\end{aligned}$$

and we compute

$$\begin{aligned}\bar{v}_4 &= \bar{v}_6 e^{v4} = 0\\ \bar{v}_3 &= v_1\bar{v}_5 = 3\\ \bar{v}_2 &= \cos(v_2)\bar{v}_3 = -1.9609\\ \bar{v}_1 &= v_0\bar{v}_4 + v_3\bar{v}_5 = -0.75680\\ \bar{v}_0 &= v_1\bar{v}_4 + 2v_0\bar{v}_2 = -7.8437.\end{aligned}$$

Hence we get

$$\frac{\partial F_1}{\partial x_1}(2,3) = -7.8437, \qquad \frac{\partial F_1}{\partial x_2}(2,3) = -0.75680.$$

To compute these derivatives the computational effort is the same as for evaluating the function.

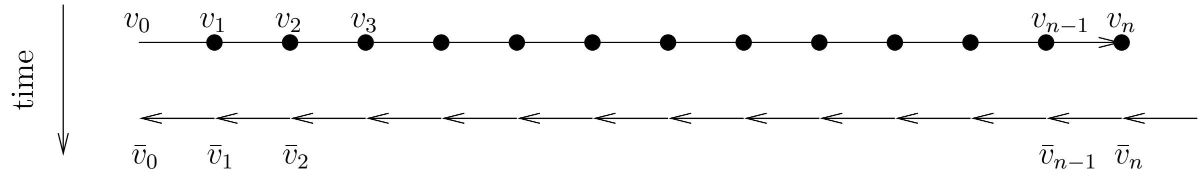


Figure 9.3: Illustration of the reverse mode: Computation of the derivative storing all intermediate results. At every black dot the intermediate result is stored. Inspired by [1].

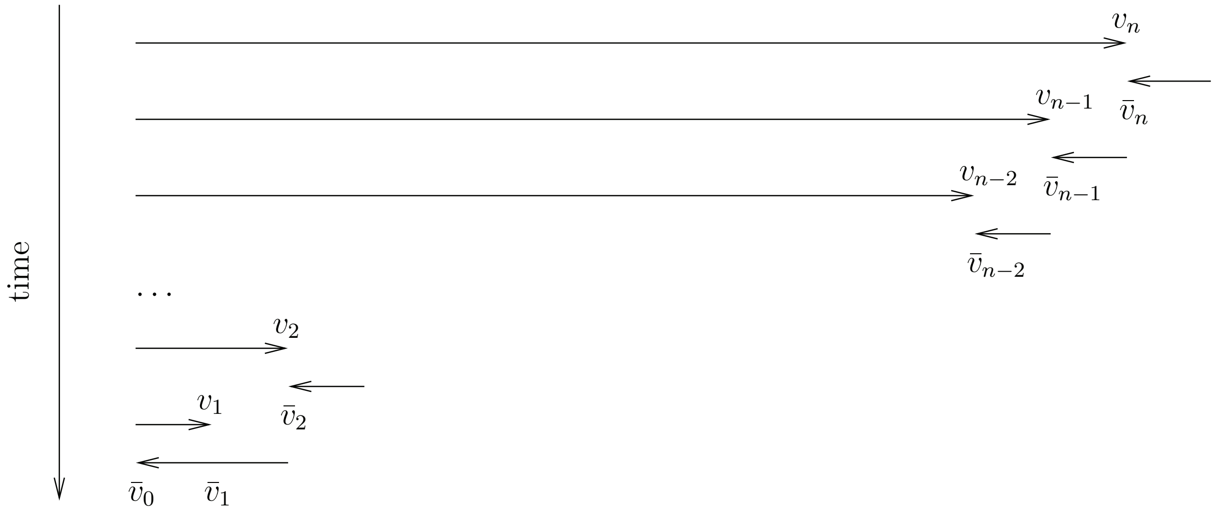


Figure 9.4: Illustration of the reverse mode: Computation of the derivative without saving intermediate results. Inspired by [1].

If we wanted to differentiate F_2 we would again have to do analogous computations as for the derivative of F_1, multiplying thus the computational cost for the differentiation by 2.

In this small example, there is not much difference in computing time between the forward and the reverse mode, but if the number of variables is large, it is preferrable to use the reverse mode because the function evaluation time would be multiplied by the number of variables. Also, if the objective function is vector-valued and has a lot of components, it is faster to compute the derivative with the forward mode than with the reverse mode.

Discussion

The automatic differentiation library we use in our implementation is called *Sacado*. It is part of the Trilinos project developed by Sandia National Laboratories [24].

The function J we want to differentiate is scalar-valued and depends on several variables (x_i, t_i, w_i). Since the computation time in reverse mode is independent of the number of independent variables, it seems profitable to use the reverse mode to compute the derivative. There, we have the choice between storing all intermediate values and recomputing them if required. However, with a computation time proportional to the square of the function evaluation time, the computation of one derivative would take too much time, and therefore the possibility of recomputing intermediate values when needed is too expensive. Thus, we would have to store all intermediate results. But preliminary tests showed that this is not feasible because the memory requirement is much too high. It seems that our function J is too involved for the reverse mode.

On the other hand, since the number of independent variables is small, the forward mode is still fast. The computation time simply gets multiplied by the number of independent variables, but this is acceptable and much less expensive than squaring the function evaluation time, and we do not have any memory issues. It is even possible to compute the Hessian of J. The function evaluation time is now multiplied by the square of the number of independent variables, which is still much less than the square of the function evaluation time, and again the memory is not a problem. The Hessian gives us the possibility to use Newton's algorithm. We therefore decided to work with the forward mode.

9.3 Particle swarm optimization

Particle swarm optimization (PSO) was introduced in 1995 by Kennedy and Eberhart [26]. The idea behind the algorithm is to simulate a swarm of animals, for example birds. The birds fly randomly around, trying to find a place with a lot of food. Their flight is guided by the point where they found most food themselves. In addition, the birds communicate with each other. Therefore, they know where the best place for food visited by any bird lies, and they are also attracted by this point. As a justification of the algorithm, Kennedy and Eberhart cite sociobiologist Wilson, who wrote in a reference about fish schooling: "In theory at least, individual members of the school can profit from the discoveries and previous experience of all other members of the school during the search for food. This advantage can become decisive, outweighing the disadvantages of competition for food items, whenever the resource is unpredictably distributed in patches" [39, p.209]. Kennedy and Eberhart took from the statement, that "social sharing of information among conspeciates offers an evolutionary advantage: this hypothesis was fundamental to the development of particle swarm optimization" [26, p.1943]. In the following algorithm the animals will be replaced by "particles".

Algorithm

We consider the minimization problem

$$\min_{\vec{q}} J(\vec{q}), \tag{9.3.1}$$

with an objective function J and optimization variables $\vec{q} = (q_1, \ldots, q_n)$. We suppose that $a_i \leq q_i \leq b_i$. For example, if we want to optimize J with respect to the feeder positions, then the variables q_i would correspond to the feeder positions x_i. In the following algorithm, a particle will denote a possible vector of optimization variables $\vec{q}$.

The algorithm has an initialization step, followed by a loop which can be split into two steps: a moving step and an updating step.

Initialization goes as follows: We choose N particles $\vec{x}_i^0, i = 1, \ldots, N$. Every particle has n components, $\vec{x}_i^0 = (x_{i,1}^0, \ldots, x_{i,n}^0)$, with $a_i \leq x_{i,j} \leq b_i$, for $j = 1, \ldots, n$. To every particle $\vec{x}_i^0$ we associate a velocity $\vec{v}_i^0$. The velocity has n components $\vec{v}_i^0 = (v_{i,1}, \ldots, v_{i,n})$, one for every variable, and it is initialized to zero. For each particle $\vec{x}_i^0$ we compute $J(\vec{x}_i^0)$. We initialize the individual best result of each particle $\mathrm{pbest}_i = \vec{x}_i^0$, and compute the best result of all particles $\mathrm{gbest} = \mathrm{argmin}\{J(\vec{x}_i^0) | i = 1, \ldots, N\}$.

We now enter the loop. For $k = 0, 1, \ldots$ we compute

- moving: for $j = 1, \ldots, n$ and $i = 1, \ldots, N$

$$v_{i,j}^{k+1} = w v_{i,j}^k + c_1 r_1 (\mathrm{pbest}_i - x_i^k) + c_2 r_2 (\mathrm{gbest} - x_i^k), \tag{9.3.2}$$

$$x_{i,j}^{k+1} = x_{i,j}^k + v_{i,j}^{k+1}, \tag{9.3.3}$$

- updating:

$$\mathrm{pbest}_i = \underset{j=0,\ldots,k+1}{\mathrm{argmin}} \{J(\vec{x}_i^j)\}, \qquad \mathrm{gbest} = \underset{i=1,\ldots,N}{\mathrm{argmin}} \{J(\mathrm{pbest}_i)\}. \tag{9.3.4}$$

The factors r_1 and r_2 are uniform random variables in the interval $(0, 1)$. They take new values for every i, j and k. The parameter w is an inertia factor which determines how fast the particles are decelerated. It is usually chosen between 0.7 and 1. The parameters c_1 and c_2 determine the weight of the attraction to the personal respectively the global best point. Usually, $c_1 = c_2 = 2$. If a particle goes beyond the boundary of a certain parameter, for example if $x_{i,j}^k + v_{i,j}^{k+1} > b_i$, then we let it rebound at the boundary. In such a case we thus set $x_{i,j}^{k+1} = 2b_i - x_{i,j}^k - v_{i,j}^{k+1}$, respectively $x_{i,j}^{k+1} = 2a_i - x_{i,j}^k - v_{i,j}^{k+1}$ if we had $x_{i,j}^k + v_{i,j}^{k+1} < a_i$. Another choice would be to set $x_{i,j}^{k+1} = b_i$, respectively a_i.

The loop is executed until either a fixed number of steps is computed, or some predefined value for J is attained, or gbest remains the same during a fixed number of steps, or the maximum velocity reaches a lower bound, etc.

One can also introduce a local best point lbest_i. Thus, for each particle $\vec{x}_i^k$ we look for the best point in the neighbourhood of the particle. To do this, we have to specify a distance $\|.\|$ and a radius ρ which define the neighbourhood of a particle. Using the local best point as a third attractor we initialize $\mathrm{lbest}_i = \vec{x}_i^0$, we compute $\vec{v}_i^{k+1}$ in the loop above by

$$v_{i,j}^{k+1} = w v_{i,j}^k + c_1 r_1 (\mathrm{pbest}_i - x_i^k) + c_2 r_2 (\mathrm{gbest} - x_i^k) + c_3 r_3 (\mathrm{lbest}_i - x_i^k), \tag{9.3.5}$$

and lbest_i is updated by

$$\mathrm{lbest}_i = \mathrm{argmin}\{J(\vec{x}_r^j) | \; \|\vec{x}_i^{k+1} - \vec{x}_r^j\| < \rho, 0 \leq j \leq k+1, 1 \leq r \leq N\}. \tag{9.3.6}$$

To illustrate the algorithm we show a small example on Figure 9.5. First we show the objective function J, which here depends on two variables $\vec{q} = (q_1, q_2)$, with q_1 in x-direction and q_2 in y-direction. The darker the color the lower is the value of J. When doing an optimization this overall picture of the function is not known.

The parameters $0.5 < w < 1$, $c_1 = 2$ and $c_2 = 2$ are set. We will work with a rebounding boundary condition. In the initialization step, the positions of two particles $\vec{x}_1$ and $\vec{x}_2$ are chosen randomly, and they are called $\vec{x}_1^0$ and $\vec{x}_2^0$ respectively. The velocities of the particles are set to zero. After the initialization we have gbest $= \vec{x}_1$.

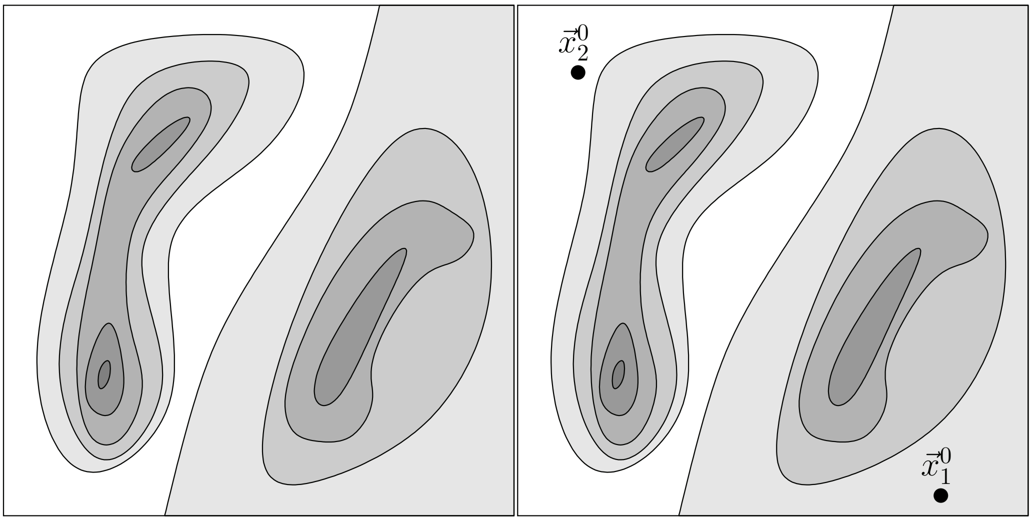
In the first step the first particle $\vec{x}_1$ is attracted by itself, and thus it does not move and $\vec{x}_1^0 = \vec{x}_1^1$. The second particle $\vec{x}_2$ is attracted by $\vec{x}_1$. Since every component of $\vec{x}_2$ is influenced by independent random variables, the particle will not necessarily move on a straight line between $\vec{x}_2^0$ and $\vec{x}_1^0$. In the example the attraction of the second component $x_{1,2}^0$ is much stronger than the attraction of the first component $x_{1,1}^0$. At the end of the first step we have $\text{pbest}_1 = \vec{x}_1^0$, $\text{pbest}_2 = \vec{x}_2^1$, and gbest $= \vec{x}_2^1$.

The particle $\vec{x}_1$ is now attracted by $\vec{x}_2^1$. Again the two components of the tuple are attracted differently. In the example the random variable determining the second component $\vec{x}_{1,2}^2$ is greater than 0.5, and thus $\vec{x}_1$ is going beyond $\vec{x}_2^1$ in the y-direction. The second particle $\vec{x}_2$ is attracted by its actual position, but since its speed is nonzero it is still moving along the same direction as in the first step, by inertia. Now we have $\text{pbest}_1 = \vec{x}_1^0$, $\text{pbest}_2 = \vec{x}_2^1$, and gbest $= \vec{x}_2^1$.

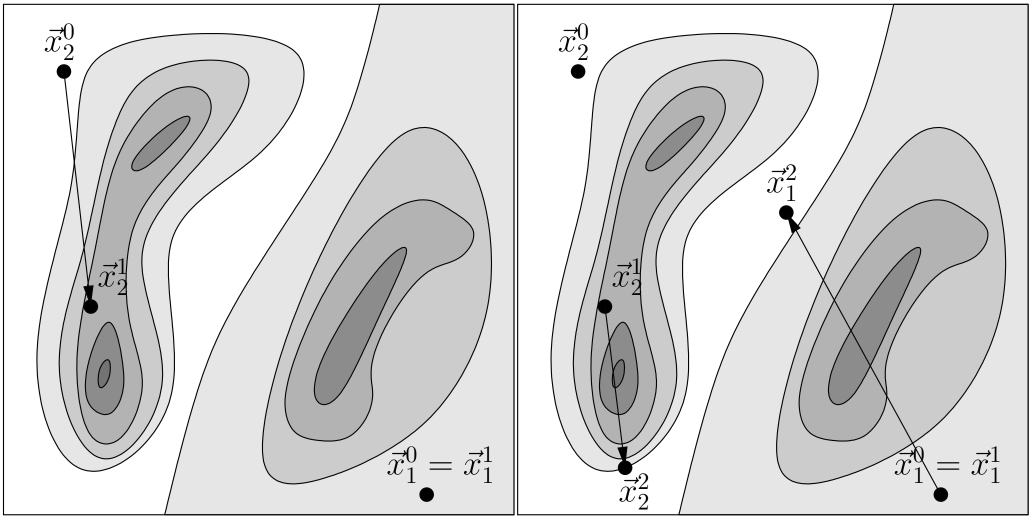
For the third step the possible outcomes can be very different. The first particle $\vec{x}_1$ is now attracted by $\vec{x}_1^0$ and by $\vec{x}_2^1$, and it will also move along the previous direction due to the inertia. In the example the particle is moving to the left, towards $\vec{x}_2^1$, but it could also have moved to the top, to the bottom or to the right, or not at all. The second particle $\vec{x}_2$ is only attracted by $\vec{x}_2^1$, which gives a smaller area of possible destinations for this third step. In the example the particle is moving to the top, towards $\vec{x}_2^1$. We have now $\text{pbest}_1 = \vec{x}_1^3$, $\text{pbest}_2 = \vec{x}_2^3$, and gbest $= \vec{x}_2^3$.

In the fourth step the particle $\vec{x}_1$ is only attracted by $\vec{x}_2^3$, and since the velocity of the previous step is pointing approximately to this point, using the rebounding boundary, the particle in the example comes very close to $\vec{x}_2^3$. The second particle $\vec{x}_2$ is attracted by itself and is thus only moving due to the inertia. After the fourth step we have $\text{pbest}_1 = \vec{x}_1^4$, $\text{pbest}_2 = \vec{x}_2^3$, and gbest $= \vec{x}_2^3$. The velocities are now quite small. This does not forcibly imply that the algorithm is going to converge soon, but if the particles are close to the actual global best result and their velocity is low, the probability that local minima far away from the actual global best result will be found is low.

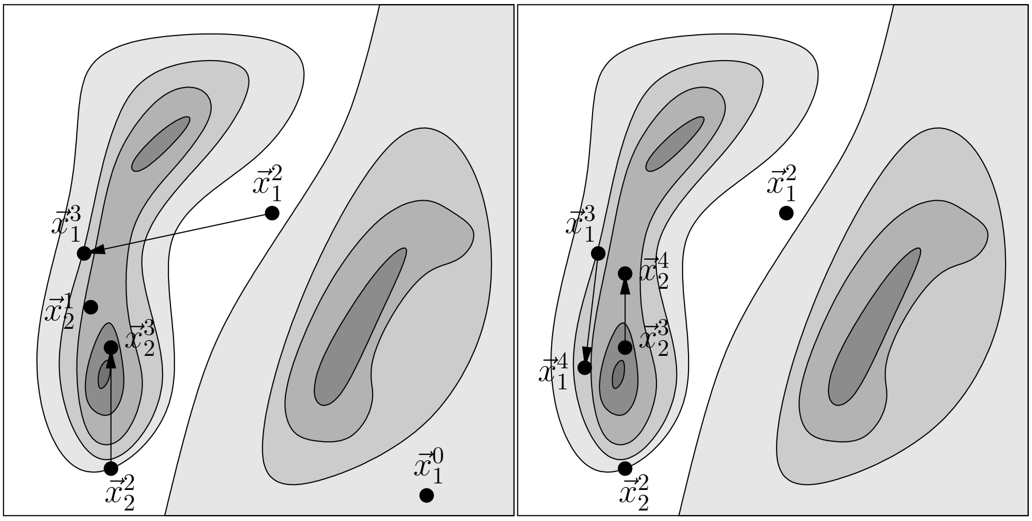
We note that the evolution of this algorithm depends very much on the random numbers that are used at every step. Using the same parameters, the algorithm could have led to a completely different result than the one given in this example, even if we start from the same initial condition.



Objective function (left) and initialization (right).



Step 1 (left) and 2 (right).



Step 3 (left) and 4 (right).

Figure 9.5: Example of PSO for a two-dimensional objective function and two particles.

9.4 Comparison of Newton's method and particle swarm optimization

The purpose of both Newton's method and PSO is to solve an unconstrained optimization problem, but the way the two methods work are completely different.

Newton's method is a deterministic optimization method, i.e., given a starting point the algorithm will always perform the same computation and arrive at the same result. In Newton's method we have to choose a starting point which is close to the solution of the problem. If we can do this, the methode converges quickly to the solution. But to make a good choice we need to have a good idea of how the objective function looks like. For a convex objective function, Newton's method always converges to the solution, but for more complicated objective functions the initial point is very important for convergence. When our starting point is far from the solution, we might not even get close to the minimizer, depending on the objective function. We say that Newton's method has good local convergence properties, but it is a bad global optimization algorithm. In addition, the objective function has to be twice differentiable, otherwise the method cannot be applied.

PSO, on the other hand, is a stochastic optimization method. This means that some parts of the computation are stochastic, and therefore given a starting point the algorithm will in general perform different computations and will maybe not arrive at the same result. In PSO, we have to choose a whole population of initial points, and convergence to a solution is quite slow. But the method tries a broader variety of candidate solutions, and even if the solution is far from all the initial points one can hope that the algorithm reaches it. Thus for global optimization, without having an idea of where to look for the solution, PSO is a good choice. By contrast, if we want to do a local search, PSO is not as efficient as Newton's method. PSO can be applied to discontinuous objective functions.

A possible way to attack an optimization problem with a difficult objective function is therefore to do a global search by PSO, not forcibly until convergence. Some of the best points found can then be used as starting points in Newton's method, in the hope that one of them is close to the minimum. This is the strategy that we will use when optimizing the feeder positions.

Chapter 10

Numerical results

Here we present some numerical results concerning the optimization problems. The aim of this chapter is not to find optimal solutions to a specific aluminium cell. It is rather to investigate the optimization of a realistic test case, in order to draw conclusions that apply to any reasonably similar cell. We start by comparing different formulations of the objective function. Then, we try to optimize the position of the feeders. In Section 10.3 we show results obtained when optimizing the load weight of the feeders. We end the chapter with a discussion about the final time for the simulation that we have to choose in order to get useful results.

10.1 Comparison of the norm for the objective function

In Section 8.3 we decided that our objective function J would be computed by

$$J = \frac{1}{\Delta T|\Omega_{\text{el}}|} \int_{T-\Delta T}^{T} \int_{\Omega_{\text{el}}} \left(c_h(\vec{x})(t) - \bar{c}_h(t)\right)^2 d\Omega dt. \tag{10.1.1}$$

We also mentioned that instead of the $L^2(\Omega_{\text{el}})$-norm of $c_h - \bar{c}_h$ over space we could use some other norm. What we need is an objective function which reaches its minimum if and only if the concentration c_h is constant over the whole bath. Hence, if we take a norm, which is positive, and which is only zero if its argument is equal to zero, we have to consider $c_h - \bar{c}_h$. Another possibility is to use the $L^2(\Omega_{\text{el}})$-norm of the gradient of c_h. This is not a norm for c_h, but it verfies our condition given above.

We will thus compute

$$J_{L^p} = \frac{1}{\Delta T|\Omega_{\text{el}}|} \int_{T-\Delta T}^{T} \|c_h(t) - \bar{c}_h(t)\|_{L^p(\Omega_{\text{el}})} dt, \tag{10.1.2}$$

for $p = 1, 2, \infty$, and also

$$J_{H^1} = \frac{1}{\Delta T|\Omega_{\text{el}}|} \int_{T-\Delta T}^{T} \|\vec{\nabla} c_h(t)\|_{L^2(\Omega_{\text{el}})} dt. \tag{10.1.3}$$

We will do this for velocity fields $\vec{u}_a$ and $\vec{u}_b$ defined in Section 5.3. The results for $\vec{u}_c$ are similar to those for $\vec{u}_a$, and therefore we will not discuss them here. In order to compare these different objective functions, we fix the position of three feeders, and we vary the fourth feeder position, x_3, over the whole length of the cell. This does not show the complete function, but it gives a good idea about how J changes when the feeder positions move, and it is easy to visualize.

On Figure 10.1 we show $J_{L^1}, J_{L^2}, J_{L^\infty}$ and J_{H^1}, computed for $\vec{u}_a$. The four functions are normalized by setting their value at $x_3 = -7.14$ [m] to 1, and they are shown at the same scale.

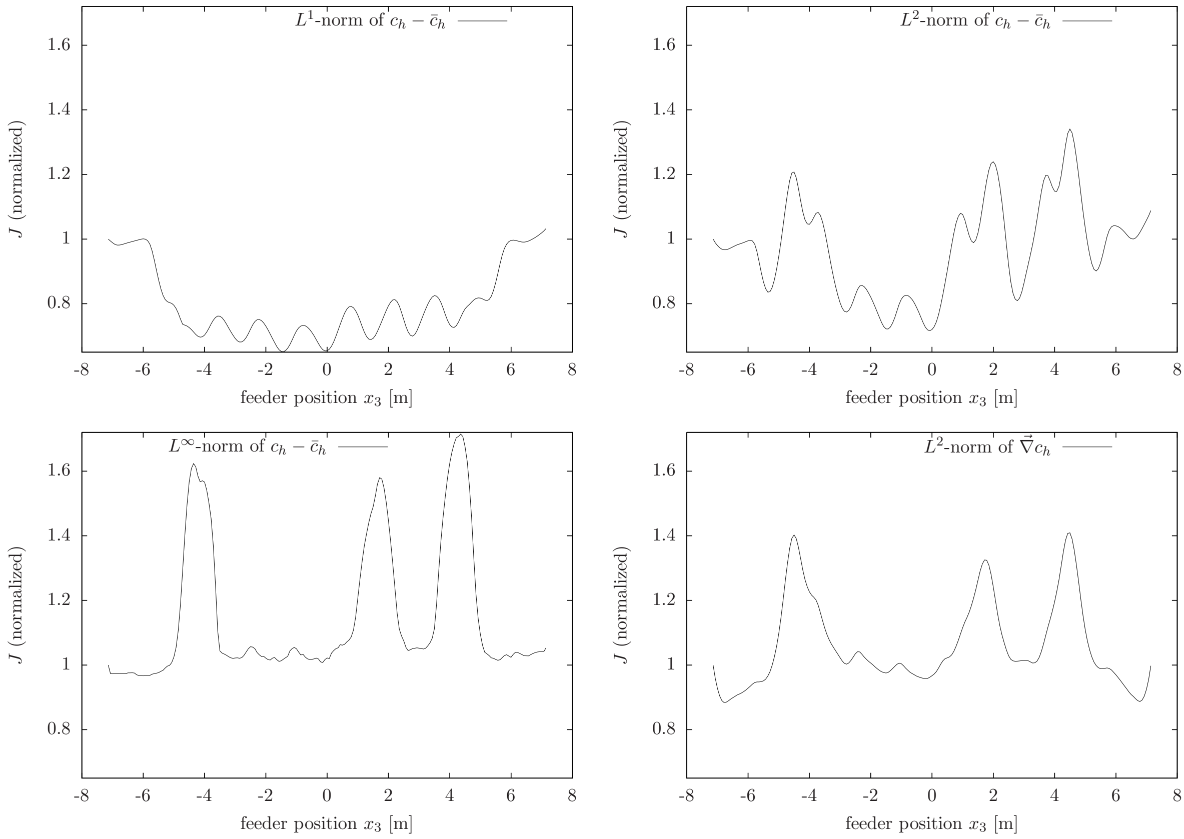


Figure 10.1: A cut of the objective function J, for $J_{L^1}, J_{L^2}, J_{L^\infty}, J_{H^1}$. Three feeding positions are fixed, the position x_3 varies over the whole length of the cell. The objective function J is normalized by setting the value at $x_3 = -7.14$ [m] to 1. For velocity field $\vec{u}_a$.

We notice that J_{L^∞} and J_{H^1} show a lot of very small variations, and they do not seem to be coninuously differentiable. Therefore, these two functions are not well suited for an opimization. The function J_{L^1} and J_{L^2} are much smoother, and their number of local minima is small. They could both be used for the optimization. We show the four functions again on Figure 10.2, here for velocity field $\vec{u}_b$. Also here J_{L^∞} and J_{H^1} are rather irregular functions with many local minima. The functions J_{L^1} and J_{L^2} are again much smoother. The number of local minima of J_{L^1} and J_{L^2} is about the same, but the minima of J_{L^2} are more distinct. Due to this, it is easier to distinguish the global minimum of J_{L^2} than the one of J_{L^1}. We therefore think that it could be advantageous to use the $L^2(\Omega_{\text{el}})$-norm to compute the objective function. Since J_{L^2} looks smooth, we can also expect that Newton's method will work well.

We note that the minimum that we find can, and in general will, be dependent on the norm with which we measure the deviation of c_h from $\bar{c}_h$. Whether the norm we just decided to use is also useful from the point of view of the application should be validated by some experiment.

We used here the same positions for the feeders as in Chapter 6. We see that the minimum is attained for $x_3 = -1.4$ [m] or $x_3 = 0$ [m]. The original setting is x_3=-1.6 [m]. Thus, if x_1, x_2 and x_4 are fixed, the optimal position of the third feeder x_3 is very close to the actually chosen position, at least according to our computation. However, when all positions are allowed to change, the result will be different as shown later on.

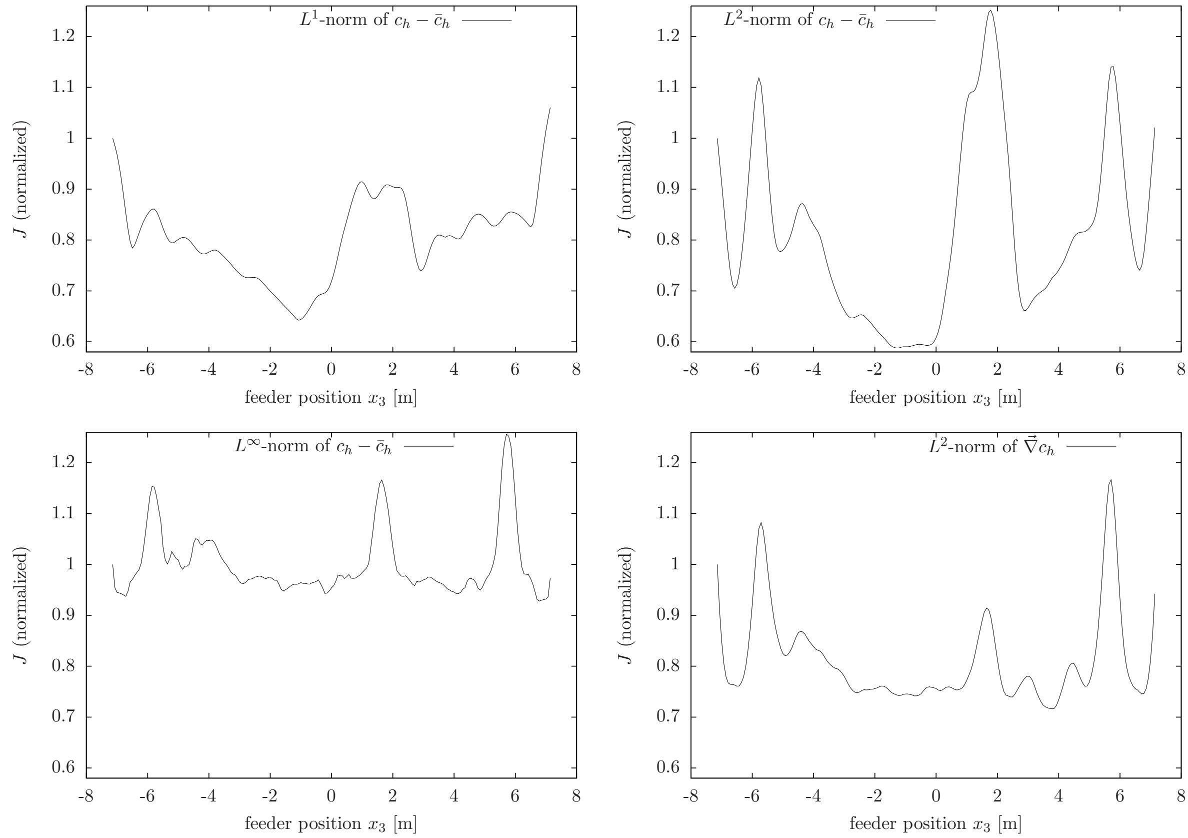


Figure 10.2: A cut of the objective function J, for $J_{L^1}, J_{L^2}, J_{L^\infty}, J_{H^1}$. Three feeding positions are fixed, the position x_3 varies over the whole length of the cell. The objective function J is normalized by setting the value at $x_3 = -7.14$ [m] to 1. For velocity field $\vec{u}_b$.

10.2 Optimization of the feeder positions

Here, we try to solve problem (8.3.8). We work with a feeding period of 64 seconds, the feeding occurs at 0, 16, 32 and 48 seconds of each feeding cycle, and the load weight of each feeder is one fourth of the total amount needed. We will use the three velocity fields $\vec{u}_a, \vec{u}_b$ and $\vec{u}_c$ defined in Section 5.3. The final time of one simulation is set to 640 seconds, which corresponds to 10 feeding cycles. A debate of whether this final time is large enough or not will be held in Section 10.4.

We minimize the objective function in two steps. First we do a particle swarm optimization, where we initialize a random population of 64 particles and compute their evolution during 14 generations. The best result found by the PSO will be used as the starting point for Newton's method.

As we could already guess from Figures 10.1 and 10.2, the objective function J is not easy to minimize. We can therefore not expect to find the global minimum of J at every trial. Hence, we execute our algorithm several times and compare the results that we obtain. Since the PSO works with random numbers, the final result depends on the sequence of random numbers that are used. When we initialize the random population for the PSO, we have to choose a seed for the random number generator. This seed completely determines the sequence of random numbers. We compute the result of the above algorithm with the seeds 1 to 10. These results are listed in Table 10.1 for $\vec{u}_a$, in Table 10.2 for $\vec{u}_b$, and in Table 10.3 for $\vec{u}_c$. In those tables, for each seed, we give the "optimal" positions that were found, and the value of the objective function. We notice that the "optimal" positions found are rarely the same for different seeds, i.e., the algorithm is not able to find the optimum for every initial condition. The values of J that are attained are not even close to each other. Thus, the convergence of the algorithm is not very good.

However, if we look closer at the positions that were found, we notice that they are very similar. We plot these positions on Figures 10.3, 10.4 and 10.5 for the different velocity fields. For each seed, reported on the ordinate, the four feeder positions x_1, x_2, x_3, x_4 are marked by different symbols to show their order. We see that for each velocity field only eight to nine different positions are found, in different orders. For $\vec{u}_a$, essentially two positions are present in every solution, and all the other positions appear several times. Using $\vec{u}_b$ and $\vec{u}_c$ there are four positions which are found by almost every optimization attempt, plus some other positions. Hence, we are not sure if the optimization algorithm converges to the optimal solution, but if we execute the algorithm several times we get a strong indication to where the optimal positions for the feeders are located.

We notice that most of the feeder positions found using $\vec{u}_a$ or $\vec{u}_c$ are almost the same. There are two more positions present when using $\vec{u}_c$, close to the boundary, which are probably found because the speed of $\vec{u}_c$ is faster close to the boundary.

Finally, for each velocity field we compute the periodic state for the best solution found and we show the distribution of the alumina concentration on Figure 10.6. The value of J for $\vec{u}_a$ when using the standard feeding of Chapter 6 is 364. The value found for the best optimization result at the periodic state is 309. When using $\vec{u}_b$ we find $J = 316$ in Chapter 6 and $J = 238$ after optimization, and for $\vec{u}_c$ we have $J = 67.6$ for the standard feeding and $J = 55.9$ at the periodic state after optimization. Hence, there is an important decrease of the value of the objective function when optimizing the position. Observe that the values at the periodic state after optimization are higher than the values found when doing the optimization and reported on Tables 10.1 - 10.3. We will comment on this in Section 10.4.

seed	x_1[m]	x_2[m]	x_3[m]	x_4[m]	J
1	5.30	-4.79	1.35	-1.48	281
2	-0.06	-2.82	5.31	-5.28	276
3	-4.79	1.35	-1.48	5.31	279
4	1.33	-2.80	-4.89	5.31	286
5	5.31	-2.82	-0.06	-5.28	277
6	5.31	-5.27	-1.50	1.34	282
7	5.31	-0.15	1.42	-5.27	287
8	-1.50	5.31	1.35	-5.28	277
9	-0.06	5.31	-5.28	-2.82	278
10	5.30	-0.14	-4.78	2.70	288

Table 10.1: Results of the optimization of the feeder positions for $\vec{u}_a$.

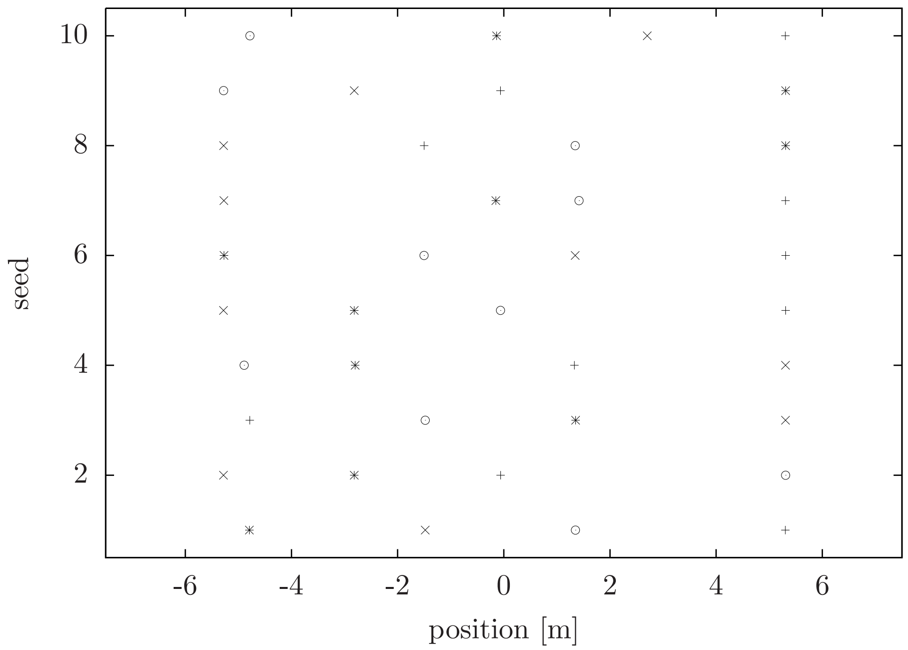


Figure 10.3: Positions of feeders found for $\vec{u}_a$. $+$ first feeder, $\ast$ second feeder, $\odot$ third feeder, $\times$ fourth feeder.

seed	x_1[m]	x_2[m]	x_3[m]	x_4[m]	J
1	3.60	-0.25	1.48	-6.57	54.9
2	1.41	-1.30	6.65	-2.85	56.2
3	-6.57	3.60	1.48	-0.25	55.1
4	1.39	-2.79	3.65	-1.20	58.2
5	1.41	6.66	-2.79	-1.23	56.0
6	4.33	-0.15	2.75	-2.67	56.0
7	-1.11	1.40	3.63	-2.78	57.7
8	3.61	-0.85	-2.78	1.42	57.9
9	-0.91	1.41	3.62	-2.77	57.6
10	-2.85	-1.31	1.41	4.23	58.9

Table 10.2: Results of the optimization of the feeder positions for $\vec{u}_b$.

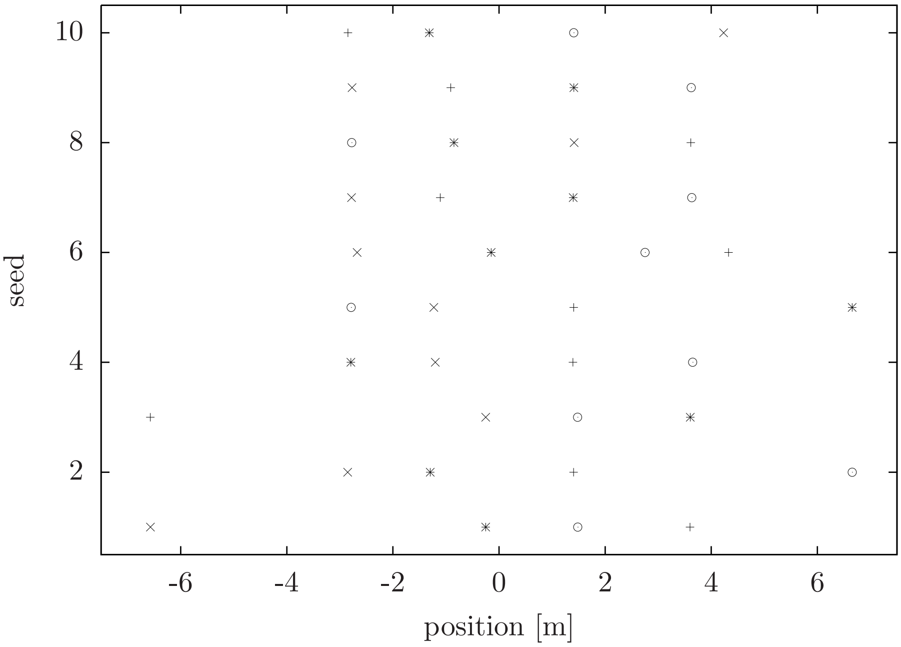


Figure 10.4: Positions of feeders found for $\vec{u}_b$. $+$ first feeder, $\ast$ second feeder, $\odot$ third feeder, $\times$ fourth feeder.

seed	x_1[m]	x_2[m]	x_3[m]	x_4[m]	J
1	5.40	1.26	-1.50	-6.96	242
2	1.24	-2.76	6.86	-5.35	251
3	5.40	-5.37	-1.51	1.26	245
4	-1.51	-5.37	6.86	1.26	238
5	-6.99	6.87	1.25	-1.50	239
6	6.86	-1.51	1.26	-5.38	237
7	1.26	-1.49	5.41	-6.98	241
8	-0.03	6.85	-1.50	-6.98	249
9	-0.10	6.86	-5.34	-2.78	255
10	6.87	-1.51	-5.38	1.26	238

Table 10.3: Results of the optimization of the feeder positions for $\vec{u}_c$.

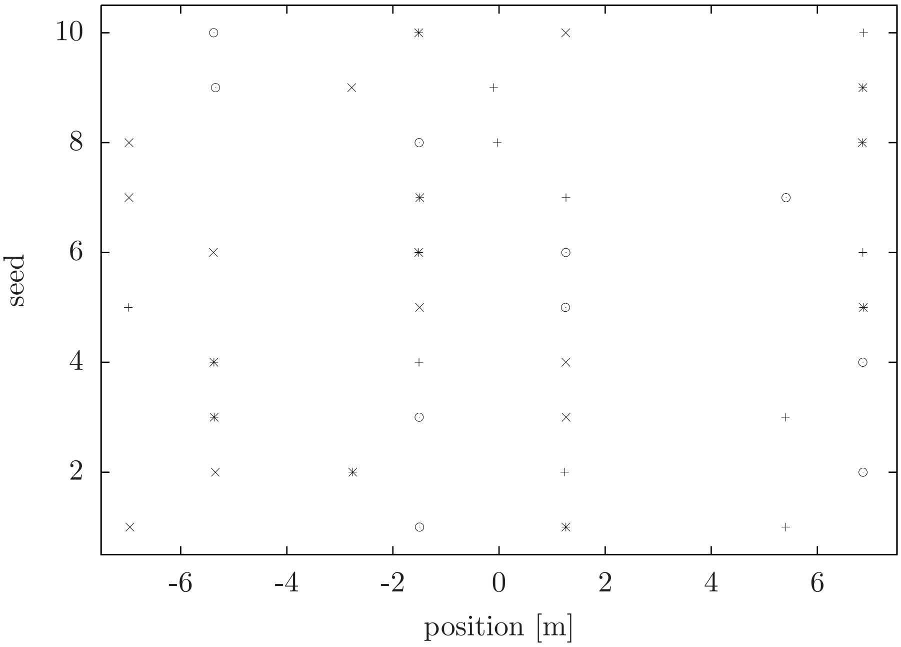


Figure 10.5: Positions of feeders found for $\vec{u}_c$. + first feeder, $\ast$ second feeder, $\odot$ third feeder, × fourth feeder.

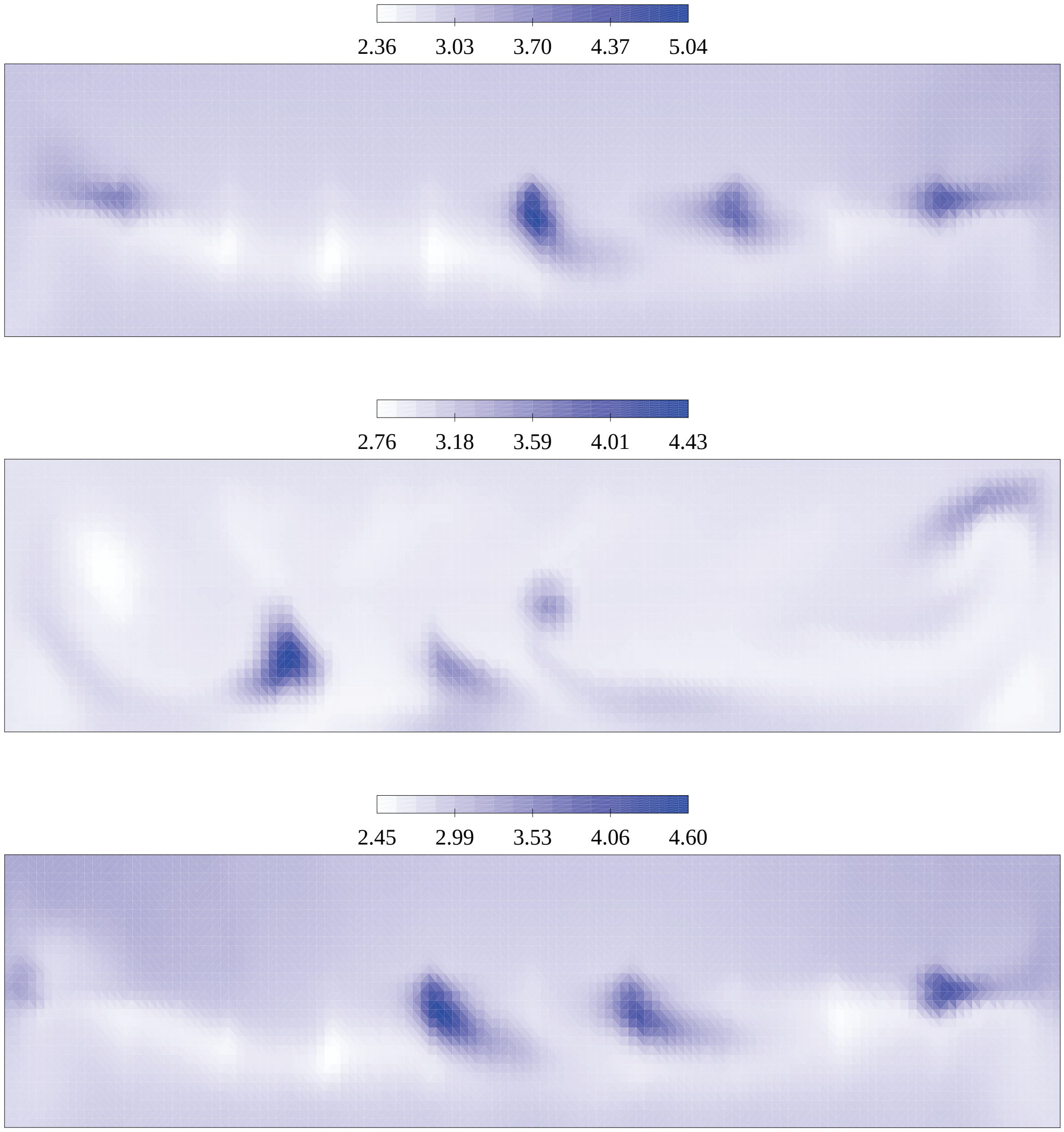


Figure 10.6: Alumina concentration in the bath for the best feeder positions using $\vec{u}_a$ (on top, with seed 2), using $\vec{u}_b$ (in the middle, with seed 1), and using $\vec{u}_c$ (on the bottom, with seed 6), at the bath - metal pad interface. View from above.

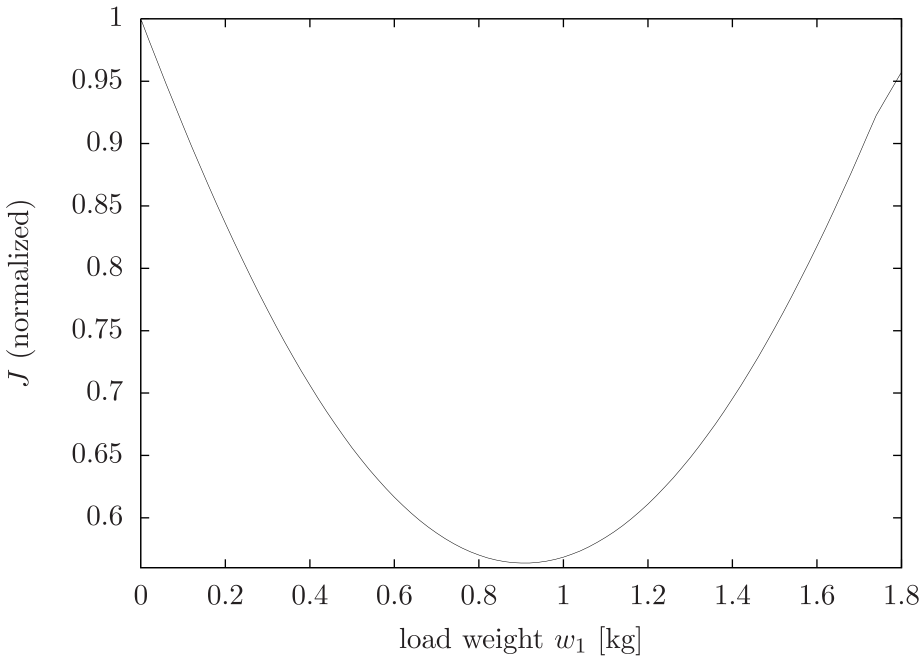


Figure 10.7: A cut of the objective function J when varying the load weight, computed using the L^2-norm for $\|.\|_b$. Two load weights are fixed, the weight w_1 varies from 0 to 1.8 [kg]. The objective function J is normalized by setting the value at $w_1 = 0$ [kg] to 1. For velocity field $\vec{u}_a$.

10.3 Optimization of the load weights

Compared to the optimization of the feeder positions the optimization of the load weights is easy. This is due to the fact that the objective function with respect to the load weights is strictly convex. At least that is what we conclude from the results. First, we show a cut of the objective function on Figure 10.7. This plot is obtained in a similar manner as the plots in Section 10.1, i.e., the weights of two feeders are fixed, and w_1, the weight of the first feeder, varies from 0 to 1.8 [kg]. The graph shown is obtained using velocity field $\vec{u}_a$, but for $\vec{u}_b$ and $\vec{u}_c$ the results are qualitatively the same. We see that the function J is smooth and convex. For a sufficiently smooth and convex function, Newton's method always converges to the minimum, independently of the starting point. We therefore choose different initial weight distributions and compute the solution, solely using Newton's method. The result is always the same. Hence we conclude that J is a globally convex function.

Finally we compute the minima for our three velocity fields $\vec{u}_a, \vec{u}_b$ and $\vec{u}_c$. We start from the normal feeder configuration of a aluminium cell, i.e., the feeding period is 64 seconds, the feeder positions are $(x_1, x_2, x_3, x_4) = (-4.4, 1.6, -1.6, 4.4)$ [m], the weight of alumina added by each feeder is one fourth of the total weight needed, and the loads are added at 0, 16, 32 and 48 seconds of each feeding cycle. The solutions are reported on Table 10.4. We show for each velocity field the optimal weights found, the value of J at the optimum, and the relative decrease of the value of J from the starting position, where the value of the objective function is denoted by J_{init}.

The optimal weight distribution between the feeders are not very different from the uniform distribution, where $w_i = 0.87$ [kg], $i = 1, \ldots, 4$. Also the decrease of J is not important, being less than 2% when using $\vec{u}_a$ and $\vec{u}_c$, and only 7.4% when using $\vec{u}_b$. Concluding from our results

velocity field	w_1 [kg]	w_2 [kg]	w_3 [kg]	w_4 [kg]	J	$\frac{J_{\text{init}}-J}{J}$
$\vec{u}_a$	0.78	0.98	0.97	0.74	343	1.9%
$\vec{u}_b$	0.72	0.63	1.09	1.03	61	7.4%
$\vec{u}_c$	0.81	0.82	1.00	0.84	304	1.1%

Table 10.4: Results of the optimization of the load weights for the different velocity fields

the configuration used in real aluminium reduction pots is therefore well chosen. We notice however from the curve on Figure 10.7 that the value of J can vary a lot when the weight distribution is changed, implying that it is important to choose good weight distributions.

We conclude that the optimization of the load weight is theoretically easy, and that it is useful to do this optimization since the variations of the objective function are important.

10.4 Discussion of the final time of the simulation

In Section 5.3, we noticed that the alumina distribution in the bath becomes periodic if the simulation time is large enough. The simulation time one has to use to get a periodic result depends on the velocity field. For velocity fields $\vec{u}_b$ and $\vec{u}_c$, the periodic state of the alumina concentration is reached after about 2000 seconds, whereas for velocity field $\vec{u}_a$ this takes about 4000 seconds.

When we optimize the distribution of the alumina concentration, we would like to use a small final time of the simulation, because we have to compute many simulations until the minimum of the objective function is found. For example, for the optimization of the feeder positions we computed 896 simulations up to the final time only for the PSO. For the computations shown in Sections 10.1 - 10.3, we therefore worked with a final time which was 10 times the length of the feeding cycle, $T = 10\Delta T$. Since we did not change the feeding period, we always had $\Delta T = 64$ [s]. Hence, the periodic state is not reached after the final time of the optimization $T = 640$ [s]. In the following, we want to evaluate if this too short final time during the optimization has an important impact or not.

We first consider the results obtained using velocity fields $\vec{u}_b$ and $\vec{u}_c$, since the periodic state is attained earlier than using $\vec{u}_a$. In Section 10.2, we computed the periodic state for the feeder positions which gave the smallest value for J. When we did the optimization, with $T = 640$ [s], the value of J was 54.9 for $\vec{u}_b$ and 237.0 for $\vec{u}_c$. At the periodic state, these values were 55.9 for $\vec{u}_b$ and 238.3 for $\vec{u}_c$, which is a difference of 1.8%, respectively 0.5%. Thus, the value of J does not change a lot after the final time T.

Another test is to look whether the value of J changes if the order of the feeders is unchanged, but the starting position is different. When we are at the periodic state of the cell, the starting position does not have any influence on the result. On Tables 10.5 - 10.6 we show the value of J at T for all possible starting positions of the best result found in Section 10.2. We notice that the values of J vary. This implies that during the optimization, these four feeder configurations were considered differently, whereas they should have been equivalent.

When optimizing the load weights in Section 10.3 we also had $T = 640$ [s]. Now we compute the solution of the optimization with $T = 1280$ [s], in order to see how much the solution changes. The result is shown on Tables 10.7 - 10.8. We notice that the weights do not change significantly, at most the difference is 2.8%, and the value of the objective function increases in the same order. But since $T = 1280$ [s] is still not the time at which the cell is in its periodic state, the optimal result could change again if T was even larger.

We conclude that the final time $T = 640$ seconds is slightly too small for the optimization if we work with the velocity fields $\vec{u}_b$ and $\vec{u}_c$. The results would be more reliable when the optimization was done for the periodic state of the cell.

x_1[m]	x_2[m]	x_3[m]	x_4[m]	J
3.60	-0.25	1.48	-6.57	54.9
-6.57	3.60	-0.25	1.48	55.1
1.48	-6.57	3.60	-0.25	55.2
-0.25	1.48	-6.57	3.60	55.1

Table 10.5: The value of the objective function J when changing the starting position of the feeding, for $\vec{u}_b$.

x_1[m]	x_2[m]	x_3[m]	x_4[m]	J
6.86	-1.51	1.26	-5.38	237.0
-5.38	6.86	-1.51	1.26	239.5
1.26	-5.38	6.86	-1.51	237.5
-1.51	1.26	-5.38	6.86	236.1

Table 10.6: The value of the objective function J when changing the starting position of the feeding, for $\vec{u}_c$.

T [s]	w_1 [kg]	w_2 [kg]	w_3 [kg]	w_4 [kg]	J
640	0.724	0.632	1.094	1.028	61.4
1280	0.719	0.621	1.103	1.034	62.2

Table 10.7: Comparison of the optimal load weights found for the final times $T = 640, 1280$ [s], using velocity field $\vec{u}_b$.

T [s]	w_1 [kg]	w_2 [kg]	w_3 [kg]	w_4 [kg]	J
640	0.809	0.824	1.003	0.841	304
1280	0.797	0.847	0.995	0.837	311

Table 10.8: Comparison of the optimal load weights found for the final times $T = 640, 1280$ [s], using velocity field $\vec{u}_c$.

Knowing that the periodic state is reached after only about 4000 seconds using velocity field $\vec{u}_a$, we now do the same checks as before for $\vec{u}_b$ and $\vec{u}_c$. The value of J for the best solution of the feeder position optimization in Section 10.2 was 276. When computing the periodic state, shown on Figure 10.6, the value of J at 4000 seconds is 309, i.e., about 12% higher than the value of J at 640 seconds. This does not forcibly imply that the result found by the optimization are non-optimal, but it is an indicator that the function J, seen as a function of the final simulation time, changes a lot after 640 seconds, and thus chances are high that the positions of the local minima move when the final time goes up.

On Table 10.9, we show the results if we change the starting position of the feeders. The differences are not enormous, but they imply that the four possibilities were treated as four different solutions, and not as four equivalent solutions, which they are.

Finally we compare the solutions found when we optimize the load weight for different final times. The results are shown on Table 10.10. Here, the weights change more importantly than for $\vec{u}_b$ and $\vec{u}_c$, the largest difference between the "optimal" weights is about 11.5%.

We see thus that the the differences are more pronounced for $\vec{u}_a$ than for the two other velocity fields. This seems reasonable since the time to reach the periodic state is higher for $\vec{u}_a$ than for the other velocity fields. Again we conclude that if we want to ensure meaningful results, the final time of the simulation has to be larger.

x_1[m]	x_2[m]	x_3[m]	x_4[m]	J
-0.06	-2.82	5.31	-5.28	276
-5.28	-0.06	-2.82	5.31	281
5.31	-5.28	-0.06	-2.82	281
-2.82	5.31	-5.28	-0.06	279

Table 10.9: The value of the objective function J when changing the starting position of the feeding, for $\vec{u}_a$.

T [s]	w_1 [kg]	w_2 [kg]	w_3 [kg]	w_4 [kg]	J
640	0.78	0.98	0.97	0.74	343
1280	0.84	0.96	0.91	0.76	355

Table 10.10: Comparison of the optimal load weights found for the final times $T = 640, 1280$ [s], using velocity field $\vec{u}_a$.

Chapter 11

Conclusion

In this second part, we wanted to optimize the feeding of alumina to the bath, in a way that the alumina concentration field in the bath would be the most uniform possible. To do this, we used the simulation described in the first part. We formulated mathematical problems with various parameters whose solutions would fulfil this aim. We also explained the optimization methods that we used, which are Newton's method with line search, particle swarm optimization, and automatic differentiation.

We wanted to optimize the feeding with respect to four parameters: the feeding times, the length of the feeding period, the load weight of each feeder, and the feeder positions. We noticed that the optimization of the feeding times was not necessary because the maximum gain was only in the order of 1%. The optimal result for the length of the feeding period is to take the shortest period possible. We noticed that the optimization of the load weight of the feeders was rather easy, because the objective function is smooth and convex. However, optimizing the feeder positions is a difficult problem since the objective function shows many local minima. At least, the objective function is smooth when computed using the $L^2(\Omega_{\text{el}})$-norm in space. For this last problem we cannot guarantee that we find the optimum.

The final time for the simulation when optimizing must be chosen sufficiently large. Otherwise, the result depends on the final time. We also noticed that the result might depend on the precise formulation of the objective function.

In this work, we did optimizations on only partially validated test cases. If an optimization has to be done, one has first to validate the velocity field, and then the simulation of the alumina feeding and dissolution. Then, the optimization parameters can be adapted to this specific case, and the final time of the simulation has to be taken such that the bath is in a periodic state after this time. The result of the optimization should be validated in a real world experiment.

Bibliography

[1] What is automatic differentiation, `http://www-sop.inria.fr/tropics/ad/whatisad.html`, retrieved in June 2010.

[2] The aluminum smelting process, `http://www.aluminumsmeltingprocess.com`, retrieved in August 2010.

[3] D. N. Arnold, F. Brezzi, and M. Fortin. A stable finite element for the Stokes equations. *Calcolo*, 21(4):337–344 (1985), 1984.

[4] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc users manual. Technical report, Argonne National Laboratory, 2008.

[5] S. Benson, L. C. McInnes, J. Moré, T. Munson, and J. Sarich. TAO user manual (revision 1.9). Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, 2007. http://www.mcs.anl.gov/tao.

[6] R. B. Bird, W. E. Stewart, and E. N. Lightfoot. *Transport phenomena.* John Wiley and Sons, New York, 2nd edition, 1960.

[7] P. B. Bochev, M. D. Gunzburger, and J. N. Shadid. Stability of the SUPG finite element method for transient advection-diffusion problems. *Comput. Methods Appl. Mech. Engrg.*, 193(23-26):2301–2323, 2004.

[8] A. N. Brooks and T. J. R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, 32(1-3):199–259, 1982.

[9] T. J. Brown, T. Bide, S. D. Hannis, N. E. Idoine, L. E. Hetherington, R. A. Shaw, A. S. Walters, P. A. J. Lusty, and R. Kendall. World mineral production 2004-08. Technical report, British Geological Survey, Keyworth, Nottingham: British Geological Survey, 2010.

[10] E. Burman. Consistent SUPG-method for transient transport problems: stability and convergence. *Comput. Methods Appl. Mech. Engrg.*, 199(17-20):1114–1123, 2010.

[11] E. L. Crow and K. E. Shimizu. *Lognormal distributions: Theory and applications.* Dekker, New York, 1988.

[12] A. Ern and J.-L. Guermond. *Theory and practice of finite elements*, volume 159 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 2004.

[13] M. Flück. Une formulation GLS pour Navier-Stokes. Internal documentation.

[14] M. Flück, A. Janka, C. Laurent, M. Picasso, J. Rappaz, and G. Steiner. Some mathematical and numerical aspects in aluminum production. *Journal of Scientific Computing*, 2008.

[15] P. Forster, V. Ramaswamy, P. Artaxo, T. Berntsen, R. Betts, D. Fahey, J. Haywood, J. Lean, D. Lowe, G. Myhre, J. Nganga, R. Prinn, G. Raga, M. Schulz, and R. V. Dorland. Changes in atmospheric constituents and in radiative forcing. In S. Solomon, D. Qin, M. Manning, Z. Chen, M. Marquis, K. Averyt, M.Tignor, and H. Miller, editors, *Climate Change 2007: The Physical Science Basis. Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, Cambridge, United Kingdom and New York, USA, 2007.

[16] L. P. Franca and T. J. R. Hughes. Convergence analyses of Galerkin least-squares methods for symmetric advective-diffusive forms of the Stokes and incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, 105(2):285–298, 1993.

[17] V. Girault and P.-A. Raviart. *Finite element methods for Navier-Stokes equations*, volume 5 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1986.

[18] N. N. Greenwood and A. Earnshaw. *Chemistry of the elements*, chapter 7. Butterworth-Heinemann, Oxford, second edition, 1997.

[19] A. Griewank and A. Walther. *Evaluating derivatives. Principles and techniques of algorithmic differentiation*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2008.

[20] K. Grjotheim, C. Krohn, M. Malinovský, K. Matiašovský, and J. Thonstad. *Aluminium Electrolysis*. Aluminium-Verlag, Düsseldorf, second edition, 1982.

[21] J. Groulier. Modelling and measurements of alumina dissolution on AP32 and AP50. Technical report, Laboratoire de Recherche des Fabrications, Rio Tinto Alcan, St-Jean-de-Maurienne, France, September 2010.

[22] B. G. Haverkamp and B. J. Welch. Modelling the dissolution of alumina powder in cryolite. *Chemical Engineering and Processing*, 37:177–187, 1998.

[23] F. Hecht. Construction d'une base de fonctions P_1 non conforme à divergence nulle dans $\mathbf{R}^3$. *RAIRO Anal. Numér.*, 15(2):119–150, 1981.

[24] M. Heroux, R. Bartlett, V. H. R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, and A. Williams. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

[25] T. Hofer and J. Rappaz. Numerical conservation schemes for convection-diffusion equations. Technical Report 21.2010, EPFL, Lausanne, Switzerland, December 2010.

[26] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks, 1995. Proceedings.*, volume 4, pages 1942–1948 vol.4, August 2002.

[27] M. B. Monagan, K. O. Geddes, K. M. Heal, G. Labahn, S. M. Vorkoetter, J. McCarron, and P. DeMarco. *Maple 10 Programming Guide*. Maplesoft, Waterloo ON, Canada, 2005.

[28] J. J. Moré and D. J. Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Software*, 20(3):286–307, 1994.

[29] J. Nocedal and S. J. Wright. *Numerical optimization.* Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006.

[30] A. Quarteroni and A. Valli. *Numerical approximation of partial differential equations*, volume 23 of *Springer Series in Computational Mathematics.* Springer-Verlag, Berlin, 1994.

[31] M. Romerio. Modèle de viscosité anisotrope pour une cuve d'électrolyse. Working paper.

[32] H. Roustan. Private communication.

[33] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7(3):856–869, 1986.

[34] G. Steiner. *Simulation numérique de phénomènes MHD: Application à l'électrolyse de l'aluminium.* PhD dissertation, EPFL, Lausanne, Switzerland, 2009.

[35] T. Tezduyar and Y. Park. Discontinuity-capturing finite element formulations for nonlinear convection-diffusion-reaction equations. *Computer methods in applied mechanics and engineering*, 59:307–325, 1986.

[36] T. Tomasino. Private communication.

[37] J. Wang and D. R. Flanagan. General solution for diffusion-controlled dissolution of spherical particles. 1. Theory. *Journal of Pharmaceutical Sciences*, 88(7):731–738, 1999.

[38] B. Welch. Aluminum production paths in the new millennium. *JOM-Journal of the minerals metals & materials society*, 51(5):24–28, May 1999.

[39] E. O. Wilson. *Sociobiology: The new synthesis.* Belknap Press, Cambridge, MA, 1975.

Curriculum vitae

I was born on 13 December 1983 in Chur, Switzerland. From 1999 to 2002 I attended the Gymnasium in Langenthal, where I obtained the qualification for university entrance. I then started my studies in mathematics at the Swiss federal institute of technology in Lausanne (EPFL). During the academic year 2004-2005 I was exchange student at Chalmers university of technology in Gothenbourg, Sweden. In 2007 I obtained a MSc in mathematical engineering. The same year I started to work for my PhD under the supervision of Professor Jacques Rappaz. My PhD is about the numerical simulation and optimization of alumina feeding and dissolution in aluminium electrolysis cells.