Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science

Developing a second order accurate level-set method for solving the one-dimensional Stefan problem

A thesis submitted to the
Delft Institute of Applied Mathematics and Applied Physics
in partial fulfillment of the requirements

for the degree of

Bachelor of Science
in
Applied Mathematics and Applied Physics

by

Corné Verburg

Delft, Nederland
November 5, 2021

BSc thesis Applied Mathematics and Applied Physics

Developing a second order accurate level-set method for solving the one-dimensional Stefan problem

Corné Verburg

Delft University of Technology

Supervisors

prof. dr. ir. C. Vuik (Applied Mathematics)

dr. ir. D. Lathouwers (Applied Physics)

Thesis committee

dr. ir. J. Dubbeldam (Applied Mathematics)

dr. ir. S. Kenjereš (Applied Physics)

November 5, 2021 Delft

Lekensamenvatting

In dit onderzoek wordt gekeken naar het Stefanprobleem. Dit probleem doet zich voor wanneer we te maken hebben met een vloeistof en een vaste stof van hetzelfde materiaal die met elkaar in contact komen. In dit geval zal ofwel de vloeistof bevriezen ofwel de vaste stof smelten. Het Stefanprobleem gaat in op de vraag met welke snelheid dit vries- of smeltproces verloopt en hoe de temperatuur binnen de vaste stof en de vloeistof is voor verschillende beginsituaties en randvoorwaarden - bijvoorbeeld een situatie waarbij de randen worden gekoeld, verwarmd of zijn geïsoleerd. Het is niet altijd mogelijk om de temperatuur en de smeltsnelheid analytisch te beschrijven. In zo'n geval maken we gebruik van numerieke computermodellen, die een schatting geven van de oplossing van het Stefanprobleem. In dit onderzoek wordt een nieuwe zogenaamde "level-set-methode" ontwikkeld en geanalyseerd, die kwadratisch nauwkeurigere oplossingen geeft voor het Stefanprobleem als de complexiteit van het model lineair wordt verhoogd.

Abstract

This work aims at finding a second order accurate level-set method which solves the Stefan problem with non-homogeneous Dirichlet boundary conditions in one dimension. The numerical accuracy of the FTCS-scheme, BTCS-scheme and Crank-Nicolson scheme for the discretization of the heat equation was considered, as well as the accuracy of the first order Upwind method, Leapfrog method and the Lax-Wendroff method for the discretization of the advection equation. A level-set method was developed using a finite volume Crank-Nicolson scheme for the discretization of the moving boundary. A second order accurate scheme for solving the advection equation was developed using Lagrange extrapolation polynomials. The moving boundary velocity was estimated using second order Lagrange polynomials. The developed method was found to be second order accurate for a specific range of ratios between time step size and spatial step size.

Contents

Introduction 1

Nomenclature 3

1 The Stefan problem 5
1.1 The heat equation . . . 5
1.2 Stefan problem . . . 6
1.3 Formulation of the investigated problem . . . 8

2 Introduction to numerical methods 11
2.1 General idea of partial differential equations . . . 11
2.2 Finite difference schemes . . . 12
2.3 Discretizing the heat equation . . . 13
2.4 Discretizing advection equations . . . 14
2.5 Lagrange interpolation polynomials . . . 15
2.6 Error analysis . . . 16

3 The finite volume method 19
3.1 Fundamental principles of the finite volume method . . . 19
3.2 Finite volume method applied to the heat equation . . . 20
3.3 Comparison with other discretization techniques . . . 21

4 The level-set method 23
4.1 The advection equation . . . 23
4.2 Theory of the level-set method . . . 23

5 Choosing numerical schemes for developing a level-set method 27
5.1 Choosing a numerical scheme for solving the heat problem . . . 27
5.2 Choosing a numerical scheme for solving the advection problem . . . 33

6 Discretization 39
6.1 Considered Stefan problem . . . 39
6.2 Space and time discretization and definitions . . . 39
6.3 Finding the front velocity and its extension . . . 40
6.4 Discretization of the advection problem . . . 43
6.5 Discretization of the heat problem . . . 44
6.6 Discretizing the numerical error . . . 46

7 Numerical results and discussion 47
7.1 Heat problem without moving boundary . . . 47
7.2 Advection problem with analytical velocity . . . 50
7.3 Stefan problem . . . 51

8 Conclusion 57

Acknowledgements 59

References 61

Appendix 63

Introduction

One of the most promising and relatively sustainable energy sources is nuclear power. This form of energy can be generated in several classes of nuclear reactors, such as molten salt nuclear reactors. In these reactors, one of the key elements is the the so-called freeze-plug. This freeze-plug is a valve made of frozen salt, meant to melt when it is necessary to drain the core of the reactor. It is important to understand the melting process which happens at the freeze-plug. If the freeze-plug melts too fast, the reactor shuts down for no reason. However, if it melts too slow, the safety of the reactor is in danger. Therefore, an accurate mathematical description of this melting process is very useful in this situation.

Fortunately, there exist mathematical ways to describe melting processes. Using the heat equation and certain physical conditions, we can describe by means of partial differential equations how fast and at which positions a solid is melting. Such a system of partial differential equations in which phase change plays a role is called a Stefan problem. This physical-mathematical problem was first introduced by Josef Stefan, a Slovenian physicist in the late nineteenth century. As Stefan found out, in many cases there does not exist an analytical solution to this solid-liquid phase change problem. Therefore, numerical simulations are often necessary to predict the behaviour of melting solids.

A well-known class of methods which is often used to solve Stefan problems numerically is the so-called level-set method. This method used a function which assigns to the points in one phase - in this work the solid phase - of the Stefan problem the negative distance to the moving boundary and to the points in the other phase - in this work the liquid phase - the positive distance to the moving boundary. The position of the moving boundary is therefore implicitly stored in the definition of the level-set function. Using the advection equation, the evolution of the level-set function, and therefore also of the moving boundary, can be described. The level-set method needs, as all numerical methods a certain discretization to be used.

The aim of this thesis is to find a second-order accurate numerical solution to the Stefan problem with non-homogeneous Dirichlet boundary conditions using the level-set method combined with the finite volume method. This work is divided into eight chapters. In the chapters 1, 2, 3 and 4 the Stefan problem and the used numerical methods are introduced to the reader to create a basic understanding of the problem and the solving methods. In chapters 5, 6, 7 and 8 this theory will be put into practice and the obtained results and conclusions will be presented there.

The report will be presented in the following structure. In Chapter 1 the Stefan problem is introduced to the reader. It is explained how the heat equation and Stefan condition can be derived mathematically and how a common Stefan problem is defined. Besides that, the so-called "similarity solution" is presented. The aim of this chapter is to give the reader a good understanding of the subject of this thesis. Chapter 2 gives an introduction into the used numerical methods. Concepts like Partial Differential Equations (PDEs), finite difference schemes and error analysis are introduced in this chapter, and some of the most common numerical schemes such as Backward and Forward Euler, Crank-Nicolson and Leapfrog are presented here, since these will play a crucial role in the rest of this work. In Chapter 3, the Finite Volume Method (FVM), which plays a central role in solving the heat equation, is described. In Chapter 4, information is provided about the level-set method, a useful mathematical way to implicitly describe the position of the moving boundary which we have to deal with.

Hereafter, in Chapter 5, the theory is applied on two numerical problems - the heat problem and the advection problem, in order to select two numerical schemes which are second-order accurate in space and time. These schemes are used to discretize the Stefan problem with a level-set approach. This discretization is described in Chapter 6. The results of this discretization are presented in the following chapter, Chapter 7. Lastly, in Chapter 8, conclusions will be drawn from the results and the limitations of the developed level-set method will be discussed. Furthermore, some suggestions for further research will be given.

Nomenclature

The following list describes the most important symbols and constants used in this work. Note that this list is not complete; if other variables are used, they will be introduced in the text.

Variables

$\mathbf{F}$	Heat flux [m^2/s]
ϕ	Signed distance function [m]
Q	Heat [J]
s	Position of the moving boundary [m]
T	Temperature [K]
t	Time [s]
v	Velocity of the moving boundary [m/s]
x	Position [m]

Model variables

Δt	Time step size [s]
Δx	Spatial step size [m]
l	Length of the domain [m]
M	Number of spatial steps [-]
N	Number of time steps [-]
s_0	Starting position of the moving boundary [m]
t^n	Time step n [s]
x_i	Grid point i [m]

Physics constants

α	Thermal diffusivity [m^2/s]
c	Specific heat capacity [J/(kg· K)]
k	Thermal conductivity [W/(m·K)]
L	Specific latent heat [J/kg]

Spaces

$\mathbb{N}$	The natural numbers
$\mathbb{R}$	The real numbers

1 The Stefan problem

This chapter aims to present the most important concepts which are necessary to understand the Stefan problem, the main subject of this work. In section 1.1, the heat equation, which plays an important role in the Stefan problem, will be derived. In the following section (1.2), the Stefan problem is presented and the Stefan condition is derived. Finally, in section 1.3, the specific Stefan problem investigated in this work is presented and an analytical solution to this problem is described.

1.1 The heat equation

In this subsection, we will derive the heat equation, which plays an important role in the formulation of the Stefan problem. The heat equation is a partial differential equation which describes the diffusion of heat as a function of space and time. We follow the derivation of the problem as presented in [9].

As we know, heat transport is an important mechanism in physics which is driven by a difference in temperature between a hotter body and a colder one. To describe the distribution of heat in space over time, we consider an open and piece-wise smooth region $\Omega \subset \mathbb{R}^n$ with boundary $\partial\Omega$. Because energy is conserved, we know that the rate of change of the total heat should be equal to the net flux through the boundary $\partial\Omega$. This leads to the following equations.

$$\rho c \frac{d}{dt} \int_\Omega T \, dV = -\int_{\partial\Omega} \mathbf{F} \cdot \boldsymbol{\nu} \, dS = \int_\Omega -\nabla \cdot \mathbf{F} dV \tag{1.1}$$

In this equation, ρ is the materials density, c is its specific heat capacity, T is the temperature function in the region Ω, $\mathbf{F}$ is the the flux density and $\boldsymbol{\nu}$ is the unit normal vector pointing outwards. The last equality is derived using Gauss's theorem under the condition that the flux density function $\mathbf{F}$ is continuously differentiable on the domain Ω. If we assume that the temperature distribution T and its temporal derivative $T_t = \frac{\partial T}{\partial t}$ exist and are continuous, we can use Leibniz' rule for differentiation under the integral sign, which leads to the following equality.

$$\rho c \int_\Omega \frac{\partial T}{\partial t} dV = \int_\Omega -\nabla \cdot \mathbf{F} \, dV \tag{1.2}$$

Now we want to rewrite this function in a more useful form. To do this, we first introduce an important theorem in analysis. A proof of this theorem can be found in [9].

Theorem 1.1. Let $f : \Omega \to \mathbb{R}$ be a function such that $f \in C(\Omega)$, where $\Omega \in \mathbb{R}^n$. If $\int_V f \, dV = 0$ for all test volumes $V \subset \Omega$, then $f \equiv 0$ on Ω.

Now, we write equation 1.2 in the following form:

$$\int_\Omega (\rho c \frac{\partial T}{\partial t} + \nabla \cdot \mathbf{F}) \, dV = 0 \tag{1.3}$$

But then we see that we can apply theorem 1.1 to the equation in the integral. This leads to the following equation, where we abbreviated $\frac{\partial T}{\partial t}$ as T_t.

$$T_t = -\frac{1}{\rho c} \nabla \cdot \mathbf{F} \tag{1.4}$$

So, in order to know how the temperature changes over time, we need to express $\mathbf{F}$ as a function of the temperature in our domain. Luckily, we know that for most materials in usual conditions, we can apply Fourier's law, which states that the heat flux is proportional to the negative gradient of the temperature:

$$\mathbf{F} = -k \nabla T \tag{1.5}$$

In this equation, k is the material's thermal conductivity. Combining the previous equations 1.4 and 1.5, we obtain the following expression for the rate of change of the temperature.

$$T_t = \frac{1}{\rho c} \nabla \cdot (k \nabla T) \tag{1.6}$$

This equation can be simplified if we assume that the thermal conductivity k of the material is constant for all temperatures T. We then obtain

$$T_t = \alpha \nabla^2 T \tag{1.7}$$

In this equation, we define $\alpha = \frac{k}{\rho c}$. Equation 1.7 is the equation which will be referred to as the heat equation in the rest of this report.

1.2 Stefan problem

This subsection aims to present the Stefan problem, the subject of research of this work. The Stefan problem is a so-called "free-boundary problem", because the system contains two sub-domains with an unknown boundary position between the two sub-domains. A typical Stefan problem has the following properties: (1) The heat distribution in and the heat transfer between the two phases can be described by (partial differential) equations, (2) there exists a distinct interface between the two phases, which are distinguishable from each other and (3) the temperature of the interface is a priori known. In this report, we will restrict ourselves to the one-dimensional form of this physics-mathematical problem. In this section, we follow the theory as described in [8].

1.2.1 Problem setting

We consider a one-dimensional closed domain $\Omega = [0, l]$, where l is the length of the domain. Initially, the domain is divided in two parts: one part is in liquid state and the other in solid state. The position of the interface which separates the two phases is denoted by $s(t)$, so its position is only a function of the time t. The domain of the liquid phase is denoted by $\Omega_L = [0, s(t))$ and the domain of the solid phase is given by $\Omega_S = (s(t), l]$. The temperature at each point $x \in \Omega$ at time $t \geq 0$ is denoted by $T(x, t)$. Note that the temperature at the interface $s(t)$ is given by $T(s(t), t) = T_m$, where T_m is the melting temperature of the solid, which is constant. A graphical representation of this problem setting is given by Figure 1.

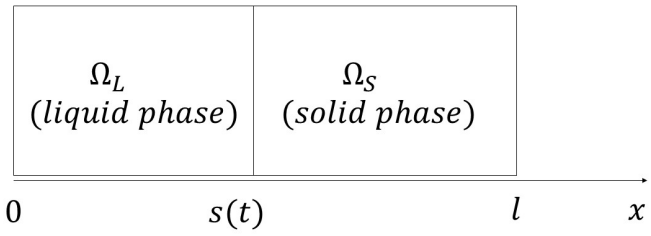


Figure 1: A graphical representation of the solid phase Ω_S, the liquid phase Ω_L and the evolving boundary $s(t)$.

In this report, we are interested in how the temperature distribution changes in time, and especially how the interface moves in time. First of all, we formulate the (one-dimensional) heat equations for this problem, using Eq. 1.6.

$$\frac{\partial T}{\partial t}(x,t) = \frac{1}{\rho_L c_L}\frac{\partial}{\partial x}(k_L \frac{\partial T}{\partial x}) \qquad x \in \Omega_L(t) \tag{1.8}$$

$$\frac{\partial T}{\partial t}(x,t) = \frac{1}{\rho_S c_S}\frac{\partial}{\partial x}(k_S \frac{\partial T}{\partial x}) \qquad x \in \Omega_S(t) \tag{1.9}$$

In this equation, ρ_L, c_L and k_L denote respectively the density, the specific heat capacity and the thermal diffusivity of the liquid phase and ρ_S, c_S and k_S denote the density, the specific heat capacity and the thermal diffusivity of the solid phase. In this section, we further assume these quantities to be constant in time and space. This leads to the following simplification of the heat equations.

$$\frac{\partial T}{\partial t}(x,t) = \alpha_L \frac{\partial^2 T}{\partial x^2}(x,t) \qquad x \in \Omega_L(t) \tag{1.10}$$

$$\frac{\partial T}{\partial t}(x,t) = \alpha_S \frac{\partial^2 T}{\partial x^2}(x,t) \qquad x \in \Omega_S(t) \tag{1.11}$$

In this equation, we defined the thermal diffusivities $\alpha_L = \frac{k_L}{\rho_L c_L}$ and $\alpha_S = \frac{k_S}{\rho_S c_S}$.

1.2.2 The Stefan condition

In order to describe the evolution of the interface position and the heat distribution, we need to find a condition on the moving boundary. An extensive derivation of this condition can be found in [9]. In order to find such a condition, we first observe that, because of physical reasons, the temperature is continuous at the solid-liquid-interface $s(t)$:

$$\lim_{x \to s(t)^+} T(x,t) = \lim_{x \to s(t)^-} T(x,t) = T_m \tag{1.12}$$

We assume that the solid is changing its phase, which means the interface $s(t)$ is moving in positive x-direction. We expect that $T \geq T_m$ in the liquid phase and $T \leq T_m$ in the solid phase. We consider the interface with area A (for example the interface of a disk) of the moving boundary at two times: t_0 and $t_1 > t_0$. Between these two times, a certain amount of solid melts and the following quantity of heat is released.

$$Q = A(s(t_1) - s(t_0)) \times \rho L \tag{1.13}$$

where Q is the released heat and L is the specific latent heat of the solid and ρ is the density of the liquid and the solid. This heat is provided by diffusion, because we assume there is no heat source or sink present. We write Fourier's law for this problem.

$$\mathbf{F_i} = -k_i \frac{\partial T}{\partial x} \tag{1.14}$$

where $\mathbf{F_i}$ denotes the heat flux of the liquid for $i = L$ and the heat flux of the solid for $i = S$. k_L is the thermal diffusivity of the liquid and k_S is the thermal diffusivity of the solid. Since the heat necessary for phase change must be provided by diffusion, we can write this the following equation.

$$Q = \int_{t_1}^{t_2} \int_A [\mathbf{F_1} \cdot \hat{x} + \mathbf{F_2} \cdot (-\hat{x})] \, dA d\tau = \int_{t_1}^{t_2} \int_A [-k_L \nabla T_L(s(\tau),\tau) \cdot \hat{x} - k_S \nabla T_S(s(\tau),\tau) \cdot (-\hat{x})] \, dA d\tau \tag{1.15}$$

By equating the two equations found for the released heat, namely Eq. 1.13 and Eq. 1.15, dividing these equations by $t_1 - t_0$ and letting $t_1 \to t_0$, we find

$$L\rho \frac{ds}{dt}(t) = k_S \frac{\partial T}{\partial x}(x,t)|_{x \downarrow s(t)} - k_L \frac{\partial T}{\partial x}(x,t)|_{x \uparrow s(t)} \tag{1.16}$$

This equation is called the Stefan condition on the free boundary. It gives us information about the velocity $\frac{ds}{dt}$ with which the boundary $s(t)$ is moving in time. Observe that the Stefan condition is just obtained by studying the implications of energy balance.

1.3 Formulation of the investigated problem

In this study, we will consider one specific form of the Stefan's problem. This problem will be solved numerically as well as analytically. In this way, we can compare the numerical solutions in their accuracy and evaluate how suitable the level-set method is for solving this problem. In this subsection, first we formulate the Stefan problem considered in this work. After this, we give the analytical solution to this problem.

1.3.1 Investigated Stefan problem

We start the formulation of the investigated problem by making some assumptions in order to simplify our problem. As described in the previous section, we consider a one-dimensional problem on the domain $\Omega = [0, l]$. We assume without loss of generality that the temperature of the melting interface is given by $T(s, t) = 0$ for all times $t > 0$. In this report, we only consider system with non-homogeneous Dirichlet boundary conditions:

$$T(0, t) = g_L(t) \qquad t \geq 0 \tag{1.17}$$

$$T(l, t) = g_S(t) \qquad t \geq 0 \tag{1.18}$$

In this equation, $g_L(t), g_S(T)$ denote time-dependent functions which indicate the temperature value at respectively the left and the right boundary. Furthermore, we need to make an assumption about the initial state of the temperatures of the liquid and the solid. In this report, we will assume that the initial temperature distribution is piece-wise constant. This leads to the following initial temperature distribution.

$$T(x, 0) = \begin{cases} T_L > 0 & \text{if } x \in \Omega_L = [0, s(0)) \quad (1.19a) \\ 0 & \text{if } x = s(0) \quad (1.19b) \\ T_S < 0 & \text{if } x \in \Omega_S = (s(0), l] \quad (1.19c) \end{cases}$$

1.3.2 Analytical solution

For most of the boundary functions $g_L(t), g_S(t)$ the analytical solution to the problem described above is unknown. However, the analytical solutions to the Stefan problem with infinite or semi-infinite domain are known. Fortunately, it is known for a fact that if we choose the values of the solution on the infinite domain at $x = 0$ and $x = l$ as the non-homogeneous Dirichlet boundary conditions to the finite-domain problem, the finite-domain solution equals the infinite-domain solution for each $x \in \Omega$. Therefore, we can use the infinite-domain solution to construct a system with Dirichlet boundary conditions of which we can find the analytical solution.

Suppose we consider the Stefan problem with infinite domain $\Omega = \mathbb{R}$. Furthermore, we assume for the sake of simplicity that the latent heat is equal to $L = 1$ and that the thermal conductivity α is constant in space and time. As initial conditions we choose the same initial conditions as for the finite-domain problem, but we extend the constant temperatures into the infinite domains. The system of equations which we want to satisfy for this analytical solution is given by the following equations.

The liquid region	$-\infty < x < s(t)$
$T_t = \alpha_L T_{xx}$,	Heat equation for the liquid region,
$T(0,t) = T_L > 0$,	Boundary condition, $t \geq 0$
$T(x,0) = T_L$,	Initial condition
The free boundary	$x = s(t)$
$s_t = k_S T_x\|_{x \downarrow s(t)} - k_L T_x\|_{x \uparrow s(t)}$	Stefan condition
$s(0) = s_0$,	Initial position of the melting interface
$T(s(t),t) = 0$,	Dirichlet condition at the interface
The solid region	$s(t) < x < \infty$
$T_t = \alpha_S T_{xx}$,	Heat equation for the solid region
$T(x,0) = T_L < 0$	For all $t, x \geq s(t)$

A complete derivation of the solutions of the Stefan problem for this domain is given in for example [6]. We find here that for $\Omega = \mathbb{R}$, the position of the interface is given by the following equation.

$$s(t) = s(0) + 2\lambda\sqrt{t} \tag{1.20}$$

The λ in this equation is given by the solution of the following equation.

$$\lambda = \frac{\sqrt{k_S}}{\sqrt{\pi}L} \frac{T_S}{\operatorname{erfc}(\frac{\lambda}{\sqrt{k_S}})} \exp\left(-\frac{\lambda^2}{k_S}\right) + \frac{\sqrt{k_L}}{\sqrt{\pi}L} \frac{T_L}{2 - \operatorname{erfc}(\frac{\lambda}{\sqrt{k_L}})} \exp\left(-\frac{\lambda^2}{k_L}\right) \tag{1.21}$$

The temperature distribution for the infinite-domain problem is given by the following equation.

$$T_{sim}(x,t) = \begin{cases} -\dfrac{T_L \operatorname{erfc}(\lambda/\sqrt{k_L})}{2 - \operatorname{erfc}(\lambda/\sqrt{k_L})} + \dfrac{T_L \operatorname{erfc}((x - s(0))/2\sqrt{k_L t})}{2 - \operatorname{erfc}(\lambda/\sqrt{k_L})} & x \in \Omega_L(t) \quad (1.22a) \\ 0 & x = s(t) \quad (1.22b) \\ T_S - \dfrac{T_S \operatorname{erfc}((x - s(0)/2\sqrt{k_S t})}{2 - \operatorname{erfc}(\lambda/\sqrt{k_S})} & x \in \Omega_S(t) \quad (1.22c) \end{cases}$$

The solution to the infinite-domain Stefan problem is often called the similarity solution. Using this solution, we can construct the non-homogeneous Dirichlet boundary conditions for our finite-domain problem, namely:

$$g_L(t) = T_{sim}(0,t) \qquad t \geq 0 \tag{1.23}$$

$$g_S(t) = T_{sim}(l,t) \qquad t \geq 0 \tag{1.24}$$

In the following Figure 2, the temperature profiles of the similarity solution at different times is shown in the domain $\Omega = [0, 1]$ with problem parameters as described in the caption. Note that we can observe a clear jump in temperature gradient around the moving boundary (which is located at $T = 0$).

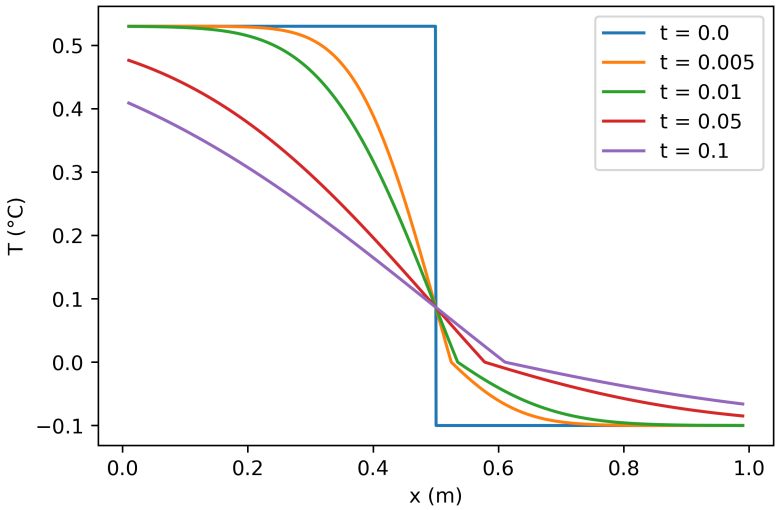


Figure 2: Temperature profiles of the similarity solution at times $t = 0$, $t = 0.005$, $t = 0.01$, $t = 0.05$ and $t = 0.1$ on the domain $\Omega = [0, 1]$. As parameters for the model, we chose $T_L = 0.53, T_S = -0.1, s_0 = 0.5, \rho = 1, L = 1, k_L = k_S = 1$

However, to find the accuracy of the numerical solutions to the Stefan problem, we will not compare the temperature profiles of the analytical and numerical solutions, but the evolution of the moving boundary $s(t)$. The boundary moves in one dimension following a square root function, as can be seen in Figure 3.

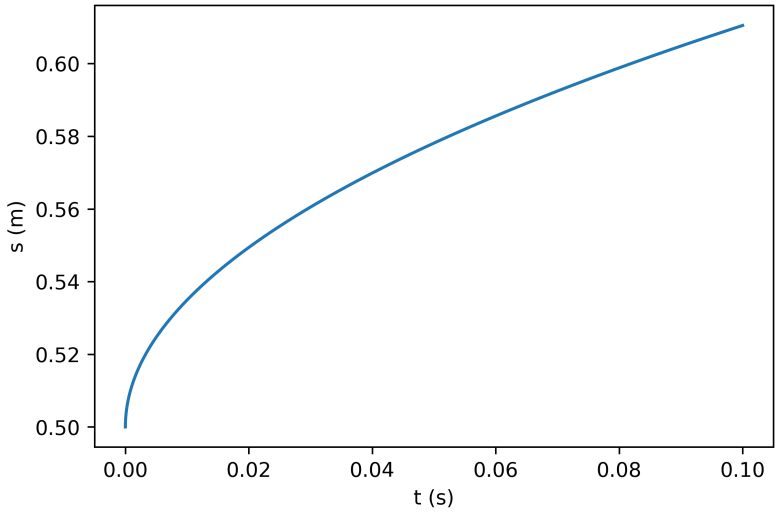


Figure 3: Position of the moving boundary over time. As parameters for the model, we chose again $T_L = 0.53, T_S = -0.1, s_0 = 0.5, \rho = 1, L = 1, k_L = k_S = 1$

2 Introduction to numerical methods

Many physical processes are described by partial differential equations. There exist several methods to find explicit analytical solutions to these equations. However, for most partial differential equations, there does not exist an explicit solution. When this is the case, numerical methods are often useful to approximate the solutions. The aim of this chapter is to describe the most important ideas behind numerical analysis and to introduce some of the most common numerical methods, which also will be used later on in this report. In section 2.1 the idea of discretizing an equation is presented. In the following section 2.2, some useful numerical methods will be discussed. In section 2.5, Lagrange interpolation, a useful method for finding values in between known values is described. Finally, the concept of numerical errors is described in 2.6.

2.1 General idea of partial differential equations

Before we turn our attention to numerical methods, we introduce some definitions. We based this section on [10]. When we consider a differential equation involving derivatives with respect to one single independent variable, we call it a *ordinary differential equation*. The *order* of an ordinary differential equation is equal to the order of the highest derivative of the dependent variable with respect to the dependent variable.

A differential equation which contains partial derivatives with respect to two or more independent variables is called a *partial differential equation* (PDE). A general classification of partial differential equations is given by the classification based on the discriminant. Suppose that f is a function of x, y and its second and first order partial derivatives are denoted by $f_{xx}, f_{yy}, f_{xy}, f_x, f_y$. Then we can write the general second order non-homogeneous partial differential equation of this function as

$$Af_{xx} + Bf_{xy} + Cf_{yy} + Df_x + Ef_y + Ff = G \tag{2.1}$$

Then, the classification of PDEs based on the sign of the discriminant is given as follows.

1. If $B^2 - 4AC > 0$, the PDE is called *hyperbolic.*
2. If $B^2 - 4AC = 0$, the PDE is called *parabolic.*
3. If $B^2 - 4AC < 0$, the PDE is called *elliptic.*

The importance of this classification is that it is closely related to the propagation of the *characteristics* of a PDE. Characteristics are the "path" of the solution domain along which information (the function value) propagates.

Approximating the solution of a partial differential equation can be done in several ways. The finite difference method and the finite volume method, two of the most common methods, are shortly described below.

1. **Finite difference methods.** This method solves a partial equation by discretizing the domain into a discrete finite difference grid and approximating the partial derivatives by finite difference approximations (as discussed in the following subsection), substituting these approximations in the PDE to obtain a discrete algebraic equation and solving the resulting algebraic system for the dependent variable.
2. **Finite volume method**. The finite volume method shows a lot of similarities with the finite difference method: the domain is discretized into a finite number of grid points. However, in the finite volume method, not the value of the function at each grid point is approximated, but the value of a volume integral around the discretized grid points. More information on this method can be found in the chapter about the finite volume method.

2.2 Finite difference schemes

The most well-known way to approximate the solution of a partial differential equation is the finite difference method. For this method, we discretize the domain into a finite number of gridpoints. The general idea of a finite difference method follows from the definition of the derivative of a smooth function f at the point $x \in \mathbb{R}$

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} \tag{2.2}$$

So, if we choose a small but finite distance Δx, we see that the quotient after this limit gives a good estimate of the derivative . The error of this estimate (i.e. the difference between the estimated value and the real function value) can be found using a Taylor expansion (assuming the function satisfies all conditions to make such an expansion)):

$$f(x + \Delta x) = f(x) + \Delta x \frac{f'(x)}{1!} + \Delta x^2 \frac{f"(x)}{2!} + \cdots + \Delta x^n \frac{f^{(n)}(x)}{n!} + R_{(n)}(x) \tag{2.3}$$

We see that in this equation the remainder term $R_n(x)$ and the higher order terms are small because of the term Δx^n. Therefore, we can approximate the first derivative as

$$f'(x) = \frac{f(x + \Delta x) - f(x)}{\Delta x} + \mathcal{O}(\Delta x) \tag{2.4}$$

We see that the difference between the "real" solution and the given approximation is of order $\mathcal{O}(\Delta x)$. We say therefore that this estimate is first order accurate. Similarly, a method of which the error is $\mathcal{O}(\Delta x^2)$, is called a second order method and so on. Higher derivatives can be approximated in a similar way. The methods which will be mentioned in the following subsections can all be derived in this way.

2.2.1 Spatial derivatives

In the Stefan problem, we need several finite difference schemes. First of all, we need to estimate the second spatial derivative of the temperature in order to solve the heat equation. Furthermore, we need to estimate the first derivative with respect to the spatial coordinate to implement the Stefan condition. Schemes often used to approximate the first spatial derivative are the forward and backward differences ($\mathcal{O}(\Delta x)$) and the central difference method ($\mathcal{O}(\Delta x^2)$). Which of these methods is the best to use, depends on the situation. The central differences scheme gives the most accurate results, but in some situations (for example the moving boundary in the Stefan condition), we can only use forward or backward differences.

$$f'(x) \approx \frac{f(x + \Delta x) - f(x)}{\Delta x} \qquad \text{Forward difference}, \mathcal{O}(\Delta x) \tag{2.5}$$

$$f'(x) \approx \frac{f(x) - f(x - \Delta x)}{\Delta x} \qquad \text{Backward difference}, \mathcal{O}(\Delta x) \tag{2.6}$$

$$f'(x) \approx \frac{f(x + \Delta x) - f(x - h)}{2\Delta x} \qquad \text{Central difference}, \mathcal{O}(\Delta x^2) \tag{2.7}$$

We also can use a second-order backward or forward scheme to approximate the first derivative. These approximations are especially useful if the data at one side of the point in which we want to approximate the derivative is not usable. This happens for example in the Stefan problem when we are close to the moving phase boundary: because of the phase change (and the jump in temperature derivative which corresponds to this change), central-difference methods will give bad results then.

$$f'(x) \approx \frac{-f(x + 2\Delta x) + 4f(x + \Delta x) - 3f(x)}{\Delta x} \qquad \text{Forward difference}, \mathcal{O}(\Delta x^2) \tag{2.8}$$

$$f'(x) \approx \frac{3f(x) - 4f(x - \Delta x) + f(x - 2\Delta x)}{2\Delta x} \qquad \text{Backward difference}, \mathcal{O}(\Delta x^2) \tag{2.9}$$

An often used method to estimate the second order spatial derivative is given by the following equation

$$f''(x) \approx \frac{f(x+\Delta x) - 2f(x) + f(x+\Delta x)}{\Delta x^2} \qquad \text{Centered difference}, \mathcal{O}(\Delta x^2) \tag{2.10}$$

2.3 Discretizing the heat equation

In this section we will consider the discretization of the simplified heat equation. This equation plays an important role in the Stefan problem, since it describes the way in which temperature evolves over time. Recall that the simplified version / where we assume that the thermal conductivity is $\alpha = 1$ of this equation was given by

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial^2 x}. \tag{2.11}$$

Discretizing this equation can be done in roughly three ways. First of all, we can use an *explicit method* (i.e. a method in which the function value at time t^{n+1} depends explicitly on the function value at time t^n). The second option is using an *implicit method* (a system of equations that needs to be solved to find the value at time t^n). The advantage of an explicit method is the lower computational power that is needed. However, explicit methods are limited by the CFL condition (which is discussed in next subsection). An implicit method on the other hand takes more computational power but does not have a time step restriction. The third choice for discretizing the heat equation by finite differences is using a combination of an implicit and an explicit method. In this way, we can combine the advantages of both methods in one method.

An explicit discretization of the heat equation is given by

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2} \tag{2.12}$$

Note that time is discretized here with a forward difference (Forward Euler) scheme and space with a central-difference scheme. This equation is therefore second order accurate in space and first order accurate in time.

An implicit discretization of the heat equation is given by the following equation. This equation is now discretized using a backward scheme in time and a central difference scheme in space, which makes this equation implicit. This method is also second order accurate in space and first order accurate in time.

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{\Delta x^2} \tag{2.13}$$

Another example of an often used method to solve the heat equation, which is second order accurate in both space and time, is the Crank-Nicoloson method. This method combines the implicit and explicit method.

$$\frac{T_i^{n+1} - T_i}{\Delta t} = \frac{1}{2}\left(\frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{\Delta x^2} + \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2}\right) \tag{2.14}$$

The Crank-Nicolson method is a special form of the so-called θ-method. In this method, the parameter $\theta \in [0, 1]$ determines the "ratio" between the explicit backward and the implicit forward scheme. This scheme is given by

$$\frac{T_i^{n+1} - T_i}{\Delta t} = \theta\frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{\Delta x^2} + (1-\theta)\frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2} \tag{2.15}$$

Note that for $\theta = 0$ this scheme equals the explicit discretization of the heat equation. For $\theta = 1/2$ the θ-method gives the Crank-Nicolson scheme and for $\theta = 1$ we see that it reduces to the implicit discretization of the heat equation.

2.4 Discretizing advection equations

In this section, we will consider the discretization of the one-dimensional linear advection equation. This equation describes the motion of a given conserved quantity which moves with a certain velocity v. The advection equation is given by

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial t} = 0 \tag{2.16}$$

where $u(x,t)$ describes the conserved quantity. In this section we will consider three methods often used to solve this method: the Upwind method, the Lax-Wendroff method and the Leapfrog method.

2.4.1 Upwind method

In the solving process of the advection equation, it is very useful to include the direction of propagation of the advection term into the method. Suppose we discretize the domain in a finite number of grid points x_i. If we consider a grid point x_i in the investigated domain, there are two directions with respect to this grid point: left and right. Suppose that the velocity v is positive. Then, the direction of advection is towards the right. In that case we call the left side of the grid point the *upwind* side and the right side the *downwind* side. In the case that the velocity v is negative, we call the left side the *downwind* side and the right side the *upwind* side.

Upwind schemes are numerical methods which change their discretization depending on the direction of propagation. For example. the first order upwind scheme is given by

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + v\frac{u_i^n - u_{i-1}^n}{\Delta x} = 0 \qquad \text{for } a > 0 \tag{2.17}$$

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + v\frac{u_{i+1}^n - u_i^n}{\Delta x} = 0 \qquad \text{for } a < 0 \tag{2.18}$$

This is often simplified by introducing the definitions $a^+ = \max(a,0)$, $a^- = \min(a,0)$ and $u_x^- = \frac{u_i^n - u_{i-1}^n}{\Delta x}$, $u_x^+ = \frac{u_{i+1}^n - u_i^n}{\Delta x}$. Then we can write

$$u_i^{n+1} = u_i^n - \Delta t[v^+ u_x^- + v^- u_x^+] \tag{2.19}$$

In order to increase the spatial accuracy of this method, more data points can be used to approximate the spatial derivative. A second-order upwind scheme is given by the following equations:

$$u_x^- = \frac{3u_i^n - ru_{i-1}^n + u_{i-2}^n}{2\Delta x} \tag{2.20}$$

$$u_x^+ = \frac{-u_{i+2}^n + 4u_{i+1}^n - 3u_i^n}{2\Delta x} \tag{2.21}$$

In order to be stable, the upwind method has to satisfy the CFL-condition. This condition states basically that information propagated from a certain grid point should not be further propagated than to its direct neighbours. We can express this in a formula as follows

$$|\frac{v\Delta t}{\Delta x} \leq 1| \tag{2.22}$$

2.4.2 Lax-Wendroff method

The second method considered in this report is the Lax-Wendroff method, which is second order accurate in space and time. This method is a so-called multi-step method. First, we approximate the value of the function $u(x,t)$ at half time steps. After this, we use the central difference method to approximate the spatial derivative using central differences. This leads to the following equations.

$$u_{i-1/2}^{n+1/2} = \frac{1}{2}(u_i^n + u_{i-1}^n) + \frac{v\Delta t}{2\Delta x}(u_i^n - u_{i-1}^n) \tag{2.23}$$

$$u_{i+1/2}^{n+1/2} = \frac{1}{2}(u_i^n + u_{i+1}^n) + \frac{v\Delta t}{2\Delta x}(u_{i+1}^n - u_i^n) \tag{2.24}$$

$$u_i^{n+1} = i_i^j - \frac{v\Delta t}{\Delta x}(u_{i+1/2}^{n+1/2} - u_{i-1/2}^{n+1/2}) \tag{2.25}$$

We can rewrite this scheme as follows:

$$u_i^{n+1} = \frac{\alpha}{2}(\alpha+1)u_{i-1}^n + (1-\alpha^2)u_i^n + \frac{\alpha}{2}(\alpha-1)u_{i+1}^n \tag{2.26}$$

where $\alpha = \frac{v\Delta t}{\Delta x}$. This method is stable if $\alpha \leq 1$ (or equivalently: $v \leq \frac{\Delta x}{\Delta t}$. Note that the numerical solution given by the Lax-Wendroff method is exactly equal to the analytical solution whenever $\alpha = 1$. This can be easily seen when we realize that the solution to the linear advection equation is given by $u(x,t) = u_0(x-at)$, where $u_0(x)$ is the initial solution at $t = 0$.

2.4.3 Leapfrog method

The last numerical method which we will discuss is the Leapfrog method. This is a second order accurate method for time as well space.

We apply this method to the advection equation. The Leapfrog method for this problem is obtained if we insert a central difference approximation of the temporal derivative and a central difference approximation for the spatial derivative into the advection equation. This leads to the following equation

$$\frac{u_i^{n+1} - u_i^{n-1}}{2\Delta t} = v\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} \tag{2.27}$$

This method is stable whenever $\frac{v\Delta t}{\Delta x} \leq 1$. Note that this system is a two-level scheme, which means that we need the information from two different time steps (n and $n-1$) to obtain the solution for step $n+1$.

2.5 Lagrange interpolation polynomials

Finite difference methods give an approximation of the function value at the gridpoints. If we want to approximate the function value on a point somewhere in between these grid points, we can use the *Lagrange interpolation polynomial*. This is the polynomial of lowest degree that assumes at each grid point x_i the corresponding function value y_i.

Given data values y_i at the locations $x_1, x_2, \ldots, x_n$, the Lagrange interpolation polynomial based on the first $j+1$ function values $u_i = u(x_i), i = 0, 1, \ldots, j$ is given by:

$$p_j(x) = \sum_{i=0}^{j} L_{i,j}(x)y_i, j = 0, 1, \ldots, n \tag{2.28}$$

In this equation, we have:

$$L_{i,j} = \frac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_j)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_j)} \tag{2.29}$$

Using this polynomial, we can approximate the function value somewhere between the data points. Furthermore, we can approximate the derivative of the function at this point by taking the derivative of the polynomial. Also note that the Lagrange polynomial can be used for extrapolation (i.e. approximating values which lie outside the given data range). Something important to note in this case is that constructing a higher order polynomial does not necessarily lead to better results, since higher-order polynomials often lead to larger oscillations.

2.6 Error analysis

In order to be able to compare the different numerical solutions of the heat equation, we need to find a measure for the error of each numerical method. Error analysis is important since it tells us where the errors come from and what we should work on to reduce them and also because it allows us to compare different numerical schemes. In this subsection, we will first take a look at consistency. After this we will consider stability and we will conclude with a description of the concept of convergence.

2.6.1 Consistency

We start with the definition of consistency.

Definition 2.1 (Consistency)**.** Let $\mathcal{F}T = \tau$ be a partial differential equation and let $\mathcal{F}^A_{\Delta x,\Delta t}T = \tau$ be a numerical scheme. We say that a numerical scheme is consistent if the discrete numerical equation tends to the exact differential equation:

$$\mathcal{F}T - \mathcal{F}^A_{\Delta x,\Delta t}T \longrightarrow 0 \qquad \text{as } \Delta x, \Delta t \longrightarrow 0 \tag{2.30}$$

A concept which can be really useful in showing consistency of a numerical method is the (local) truncation error. The (local) truncation error τ^n is defined as the error in approximating differential operators and PDEs by discrete representations like finite differences. This error is given by

$$\tau(x,t) = \mathcal{F}^A T(x,t) \tag{2.31}$$

where $\mathcal{F}^A$ is the approximation to the differential equation and $T(x,t)$ is an exact solution to the problem. This error can be analyzed by applying a Taylor expansion. For example, the local truncation error for the forward Euler method combined with the central difference method applied to the heat equation (also known as the FTCS-method) is given by the following derivation

$$\tau(x,t) = \frac{T(x,t+\Delta t) - T(x,t)}{\Delta t} - \frac{T(x-\Delta x,t) - 2T(x,t) + T(x+\Delta x,t)}{\Delta x^2} \tag{2.32}$$

$$= \frac{\partial T}{\partial t}(x,t) - \frac{\partial^2 T}{\partial x^2}(x,t) + \frac{\Delta t}{2}\frac{\partial^2 T}{\partial t^2} - \frac{\Delta x^2}{12}\frac{\partial^4 T}{\partial x^4} + \dots \tag{2.33}$$

$$= \frac{\Delta t}{2}\frac{\partial^2 T}{\partial t^2} - \frac{\Delta x^2}{12}\frac{\partial^4 T}{\partial x^4} + \dots \tag{2.34}$$

So, we see that the for the truncation error for this numerical scheme it holds that $\tau(x,t) = \mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2)$. This means that, when the time step and spatial step size go to 0, then we also know that the local truncation error goes to zero. This means that the numerical scheme in this example is consistent.

2.6.2 Stability

A problem that every mathematician has experienced in numerical modelling is that errors in numerical solutions can become uncontrolled. Therefore, an important notion in numerical analysis is the notion of stability, which means that the difference between the computed solution and the analytical solution of the discrete equation should be bounded for given spatial step size Δx when time t^n goes to infinity. In a formal definition:

Definition 2.2 (Stability)**.** Let T_i^n be the numerical solution of the partial differential equation given by $\mathcal{F}T = \tau$ at position x_i and for time step t^n. Let the spatial step size Δx and the number of spatial steps M be fixed. We say that a numerical scheme is stable if for all $n \in \mathbb{N}$:

$$(\Delta x \sum_{i=1}^{M} |T(x_i,t^n) - T_i^n|^2)^{1/2} < \infty \tag{2.35}$$

Note that we used the L^2-norm in this definition. However, we could also use other norms, for example the following (we show the L^1-, L^2- and L^{inf}-norms).

$$||e^n||_1 = \Delta x \sum_{i=0}^{M} |e_i^n| \tag{2.36}$$

$$||e^n||_2 = (\Delta x \sum_{i=0}^{M} |e_i^n|^2)^{1/2} \tag{2.37}$$

$$||e^n||_\infty = \max_{0 \leq i \leq M} |e_i^n| \tag{2.38}$$

where $e_i^n = T(x_i, t^n) - T_i^n$. The error $||e^n||$ is known as the global truncation error (or "discretization error"). A necessary condition for the stability of an explicit scheme is developed by Courant, Friedrichs and Lewy. The CFL-condition states that it is a necessary condition for the numerical solution of a PDE that the distance which is travelled of information has to be lower than the distance between two grid points. Mathematically, this can be expressed as follows

$$v \frac{\Delta t}{\Delta x} \leq 1 \tag{2.39}$$

where v is the "advection" velocity of the solution. If an explicit numerical scheme satisfies the CFL-condition, then it is stable. However, note that this is not yet enough proof for convergence, a concept which will be discussed in the following subsection.

2.6.3 Convergence

First, we give a definition of convergence.

Definition 2.3 (Convergence)**.** A numerical scheme is called convergent if the difference between the computed solution T_i^n and the exact solution $T(x_i, t^n)$ (i.e. the error $e_i^n = T(x_i, t^n) - T_i^n$) vanishes when $\Delta x, \Delta t$ vanish:

$$\lim_{\Delta t, \Delta x \to 0} |e_i^n| \tag{2.40}$$

for fixed values of the position x_i and the time step t^n

This is the most important property of a numerical scheme, but difficult to verify directly for a given scheme. However, the definitions of stability and consistency, which were shown in the previous sections, are much easier to derive. The equivalence theorem of Lax states that, if a numerical scheme is consistent and stable, then it is convergent.

3 The finite volume method

The purpose of any numerical method is to create a solvable system of algebraic equations which corresponds to the partial differential equations which determine the system. The finite volume method is a certain way of discretizing a partial differential equation which is based on discretizing the integral form of the governing equations over each control volume. An important implication of this approach is that quantities such as energy and mass are conserved for the finite volume method. The aim of this chapter is to describe this method, its use and its applications for the Stefan problem. In order to do so, in section 3.1 the fundamental principles of the finite volume method are presented. In section 3.2 we apply the finite volume method to the heat equation. In the last subsection of this chapter 3.3, we compare the finite volume method with other numerical methods such as the finite difference method.

3.1 Fundamental principles of the finite volume method

Before introducing the most important definitions and concepts for the finite volume method, we mention some of the most useful properties of this method [3]: (1) it can be used on arbitrary geometries, for example unstructured and structured grids, (2) it leads to robust schemes and (3) the numerical fluxes are locally conserved (i.e the numerical flux from one cell to its neighbour cell is conserved). Especially this last property makes the finite volume method very useful for problems where flux (of mass, heat or energy) plays a role.

Now, we will explain some theory of this method. We follow the theory from [4]. We consider a conservation law of the form

$$u_t(x,t) + \nabla \cdot \mathbf{F} = f(x,t) \tag{3.1}$$

In this equation, u is the conserved equation, $\mathbf{F}$ is some function denoting the transport mechanism of the quantity u and f expresses the "source term", the extra production of u for example due to chemical reactions. The flux function $\mathbf{F}$ is usually a function of the conserved quantity $u(x,t)$, in the heat equation this term is p.e. given by Fourier's law $\mathbf{F}(x,t) = -\nabla u(x)$.

The conservation law in the cited form is the expression of conservation of the quantity u in a certain, infinitesimal domain. This equation is equivalent to a form where we integrate over a certain temporal and spatial domain:

$$\int_K u(x,t_2)dx - \int_K u(x,t_1) + \int_{t_1}^{t_2}\int_{\partial K} \mathbf{F}(x,t)\cdot \mathbf{n_K}(x)d\gamma(x)dt = \int_{t_1}^{t_2}\int_K f(x,t) \tag{3.2}$$

In this equation, K is an arbitrary chosen sub-domain, t_1, t_2 are arbitrary times and $\mathbf{n_k}(x)$ is the unit normal vector to the boundary of K at point x, pointing outward to K. $d\gamma(x)$ is the integration symbol for the $(d-1)-$dimensional Haussdorf measure on the considered boundary.

3.1.1 Time discretization

Now, we investigate the time discretization for finite volume methods. We choose, for the sake of simplicity, a constant time step $\Delta t \in \mathbb{R}_{\geq 0}$. We start at time $t_0 = 0$. There are two ways to discretize the conservation law for time.

1. We can use a space-time finite volume discretization. This is done by integrating the conservation law over a time interval and over a space control volume. This leads to a temporal discretization.

2. The second way to implement temporal discretization is to substitute a time finite difference scheme into the conservation equation for the time derivatives. For example, an Forward Euler scheme could be used to estimate the term u_t by $u_t \approx \frac{u^{n+1}-u^n}{\Delta t}$. Other, implicit and higher-order schemes could also be used.

3.1.2 Space discretization

In order to create a spatial discretization of the conservation law, we introduce a mesh $\mathcal{T}$ of the domain Ω of $\mathbb{R}$, the investigated domain. We choose our mesh in such a way that the closure of our domain, $\bar{\Omega}$, is equal to the union of the closures of the control volumes: $\bar{\Omega} = \bigcup_{K\in\mathcal{T}} \bar{K}$. In this equation, each element $K \in \mathcal{T}$ is an open subset of Ω and is called a control volume.

We integrate our conservation law over each cell K of the mesh $\mathcal{T}$. We use the forward Euler time discretization to obtain

$$\int_K \frac{u^{n+1} - u^n}{\Delta t} dx + \int_{\partial K} \mathbf{F}(x, t_n) \cdot \mathbf{n_K}(x) d\gamma(x) = \int_K f(x, t_n) dx \tag{3.3}$$

The last step which is necessary in order to obtain a finite volume scheme, is applying a finite difference approximation to the heat flux term across the boundary of the control volumes. In order to do so, define $K|L = \bar{K} \cap \bar{L}$, with $K, L \in \mathcal{T}$. Note that this definition does not include the boundaries of Ω. Then, the exchange of the quantity between K and L $\int_{K|L} \mathbf{F}(x, t_n) \cdot \mathbf{n}_K(x) d\gamma(x)$ in the time interval $[T_n, t_{n+1})$ is approximated by some quantity $F^n_{K,L}$, which is a function of u^n_M (for explicit methods) or u^n_{n+1} (for implicit methods) or of both (for semi-implicit methods).

3.2 Finite volume method applied to the heat equation

One of the most important equations in the Stefan problem is the heat equation. This equation involves heat flux and therefore we expect good results when solving this equation using the finite volume method. In this subsection, we derive the finite volume discretization of the (simplified) one-dimensional heat equation. In this derivation, we consider constant time step size Δt and spatial step size Δx. Recall that the heat equation in one dimension (with heat diffusivity $\alpha = 1$) is given by

$$\frac{\partial T}{\partial t} - \frac{\partial^2 T}{\partial^2 x} = \frac{\partial T}{\partial t} - \frac{\partial}{\partial x}\frac{\partial T}{\partial x} = 0. \tag{3.4}$$

If we reconsider the standard definition of a conservation law, we see that in this equation $u(x,t) = T(x,t)$, $\mathbf{F}(x,t) = -T_x(x,t)$ and $f(x,t) = 0$. So, if we integrate this equation over a certain spatial domain $K = [x_{i-1/2}, x_{i+1/2}]$ (with midpoint x_i) and temporal domain $[t^n, t^{n+1}]$, with spatial step size Δx and time step size Δt we obtain

$$\int_{t^n}^{t^{n+1}} \int_{x_{i-1/2}}^{x_{i+1/2}} \frac{\partial T}{\partial t} dx dt - \int_{t^n}^{t^{n+1}} \int_{x_{i-1/2}}^{x_{i+1/2}} \frac{\partial^2 T}{\partial x^2} dx dt = 0 \tag{3.5}$$

We can interchange the integrals, which leads to the following expression for the left hand side of Equation 3.5.

$$\int_{t^n}^{t^{n+1}} \int_{x_{i-1/2}}^{x_{i+1/2}} \frac{\partial T}{\partial t} dx dt = \int_{x_{i-1/2}}^{x_{i+1/2}} \int_{t^n}^{t^{n+1}} \frac{\partial T}{\partial t} dt dx. \tag{3.6}$$

This expression can be evaluated using the fundamental theorem of calculus

$$\int_{x_{i-1/2}}^{x_{i+1/2}} \int_{t^n}^{t^{n+1}} \frac{\partial T}{\partial t} dt dx = \int_{x_{i-1/2}}^{x_{i+1/2}} T(x,t)|_{t_n}^{t^{n+1}} dx = \int_{x_{i-1/2}}^{x_{i+1/2}} T(x, t^{n+1}) - T(x, t^n) dx. \tag{3.7}$$

Now, observe that we can approximate the average temperature over a grid cell by the following expression

$$\hat{T}_i(t) = \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} T(x,t) dx \tag{3.8}$$

where $\hat{T}_i(t^n)$ denotes the approximation of the average temperature in gridcell i at time $t = t^n$. Using this definition, we can simplify the integral expression in Equation 3.7 further to

$$\int_{x_{i-1/2}}^{x_{i+1/2}} T(x,t^{n+1}) - T(x,t^n)dx = \Delta x(\hat{T}_i^n - \hat{T}_i^n) \tag{3.9}$$

where $\hat{T}_i^n$ is the approximation of the average temperature at time t^n and position x_i.

The other part of equation 3.5 can also be simplified under the assumption that the temperature in a control volume is constant over the whole control volume. This assumption becomes more valid for smaller step sizes Δx and it leads to the following equation:

$$\int_{t^n}^{t^{n+1}} \int_{x_i}^{x_{i+1}} \frac{\partial^2 T}{\partial x^2} dxdt = \int_{t^n}^{t^{n+1}} \int_{x_i}^{x_{i+1}} \frac{\partial}{\partial x}(\frac{\partial T}{\partial x})dxdt. \tag{3.10}$$

Using the fundamental theorem of calculus, this equation can again be simplified to

$$\int_{t^n}^{t^{n+1}} \frac{\partial T}{\partial x}|_{x_{i-1/2}}^{x_{i+1/2}} dx = \int_{t^n}^{t^{n+1}} \frac{\partial T}{\partial x}(x_{i+1/2},t) - \frac{\partial T}{\partial x}(x_{i-1/2},t)dt \tag{3.11}$$

We see that this equation simply is the difference in heat fluxes through the right and left boundary of our control volume integrated over time. In order to find a numerical scheme, we want to estimate the fluxes. This can be done using an explicit or implicit forward and backward difference scheme. In this case, we use an explicit central difference scheme

$$\frac{\partial T}{\partial x}(x_{i+1/2},t) = \frac{T_{i+1}^n - T_i^n}{\Delta x} \tag{3.12}$$

$$\frac{\partial T}{\partial x}(x_{i-1/2},t) = \frac{T_i^n - T_{i-1}^n}{\Delta x} \tag{3.13}$$

We use the second-order approximation that the average temperature in a gridcell is equal to the temperature in the center of the gridcell: $T_i^n = \hat{T}_i^n$. Filling this in the computed approximation in the integral form of the conservation law yields

$$\Delta x(\hat{T}_i^{n+1} - \hat{T}_i^n) = \int_{t^n}^{t^{n+1}} \frac{\hat{T}_{i+1}^n - \hat{T}_i^n}{\Delta x} - \frac{\hat{T}_i^n - \hat{T}_{i-1}^n}{\Delta x} dt \tag{3.14}$$

which can be further simplified by evaluating the integral, which leads to

$$\frac{\hat{T}_i^{n+1} - \hat{T}_i^n}{\Delta t} = \frac{\hat{T}_{i+1}^n - \hat{T}_i^n + \hat{T}_{i-1}^n}{\Delta x^2} \tag{3.15}$$

Observe that this discretization seems to be equal to the FTCS finite difference discretization over the same grid. However, in the finite volume method equation, $\hat{T}_i^n$ is not the exact temperature in a gridpoint (as it is in the FCTS), but the average temperature over a volume cell. This is a fundamental difference with the finite difference method. However, for this specific problem, both methods give the same results. Note that this may be different for other forms of the heat problem or other discretizations of time and space.

3.3 Comparison with other discretization techniques

The finite volume method differs significantly from other numerical methods such as the finite difference method or the finite element method. In this section we will restrict ourselves mainly to the differences between the finite difference methods and the finite volume method. For listing these differences, we used the theory in [3].

1. **Meaning of the results**. The first and most important difference between finite difference methods and the finite volume method is the interpretation of the results. Finite difference methods predict discrete values of the unknown, and the values of the unknown between the grid points has to be imagined by the user (for example by interpolation). The idea of the finite volume method on the other hand does not compute discrete unknown values, but the average value of a conserved variable within the chosen control volumes. Therefore, it gives a solution for the whole domain, not only for the discretization points.

2. **Differential versus integral form**. The finite difference method is based on a chosen grid where at each grid point one unknown variable has to be computed by solving one equation. Derivatives of this unknown are estimated by finite differences, which are found by Taylor expansions. Finite volume methods, on the other hand, use the integral form of an equation, which they solve by applying the divergence theorem.

3. **Conservation of unknown quantity**. One of the main advantages of the finite volume method is that it is a conserving method, i.e. the solution is conserved. The total value of a certain quantity over a control volume remains equal to the start quantity plus or minus the flux of this quantity after a certain time. This is especially useful when modelling physical problems where conservation laws play a role.

4. **Discontinuities**. Finite difference methods often have have problems whenever discontinuities in the solution arise. The finite volume method can deal much better with discontinuities, especially when the mesh is chosen in such a way that the discontinuities occur on the boundaries of a grid cell. This has to do with the fact that finite volume methods use the integral form of conservation laws, where finite difference methods make use of the differential (weaker) form of the same laws.

5. **Implementation of boundary conditions**. Note that for Neumann boundary conditions, the finite volume method is a very natural choice, since this method used the in-going and outgoing flux to update the average value of a grid cell. The Neumann boundary conditions give an exact value for the flux at the boundaries. Similarly, for Dirichlet boundary conditions, finite difference methods are often a more natural choice, since the exact value at the boundary grid points is known in this case. In order to implement Neumann boundary conditions in a finite difference method or Dirichlet boundary conditions in a finite volume method, inter- or extrapolations have to be made, which can lead to more numerical errors.

4 The level-set method

One of the most important things to deal with in the Stefan problem is the moving boundary. Because of the temperature derivative discontinuity, the unknown values around the moving boundary need a special treatment. Globally, we can divide the methods to solve the Stefan problem in (1) front-tracking methods, which compute the interface position explicitly at each time step and (2) implicit methods, in which the position is implicitly stored in some alternative way. One of the most well-known implicit methods is the level-set method, which is further investigated in this chapter. In section 4.1, the advection equation, which plays an important role in the level-set method, is discussed. In the following section 4.2, the general idea of the level-set method is discussed and a step-by-step implementation of the method is given.

4.1 The advection equation

In this section, we describe some theory which is necessary to understand the advection equation, an important equation in the level-set method. We base this section mainly on the work of Chen et al. [2]. We describe the theory of advection equations in the context of a level-set function in one dimension. Let $\Omega = \mathbb{R}$ be the domain and define the initial level-set function as follows.

$$\phi(x,0) = \begin{cases} |x - s_0| & \text{if } x < s_0 \quad (4.1a) \\ 0 & \text{if } x = s_0 \quad (4.1b) \\ -|x - s_0| & \text{if } x > s_0 \quad (4.1c) \end{cases}$$

where $s_0 \in \mathbb{R}$ is the initial value of the level-set. Note that this position is a simple straight line with negative slope -1, which crosses the x-axis at $x = s_0$. This function gives an implicit description of the position of the point s_0. We store the value where the level-set function is zero in the zero level-set $\Gamma(t) = \{x \in \mathbb{R} : \phi(x,t) = 0\}$. Note that this set (for a one-dimensional level-set function) contains only one value, which we call $s(t)$. Note that we can differentiate the level-set function at position $s(t)$ with respect to time t and that this derivative equals 0 (since $\phi(s(t)) = 0$ by definition of $s(t)$). Using the chain rule, we find

$$\frac{\partial \phi}{\partial t}(s(t),t) + \frac{ds}{dt}(t)\frac{\partial \phi}{\partial x}(s(t),t) = 0 \tag{4.2}$$

Defining the velocity of the zero level-set as $\frac{ds}{dt}(t) = v(t)$, we can write this as

$$\frac{\partial \phi}{\partial t} + v(t)\frac{\partial \phi}{\partial x} = 0 \tag{4.3}$$

This is the general form of a one-dimensional advection equation: the equation that governs the motion of a scalar field. Note that the advection velocity of the level-set function depends on the time: it is not constant. This is the partial differential equation which will be considered in this report for describing the evolution of the level-set function over time. If we can find the velocity $v(t)$ and we know how the level-set function looks at time t, we can use this advection equation to find the temporal derivative ϕ_t which can be used then to update the level-set function to the next time step.

4.2 Theory of the level-set method

In this section, we shortly discuss the most important concepts and ideas behind the level-set method. This method provides a means for solving moving boundary Stefan problems, such as the Stefan problem. We will lay out this method applied to the Stefan problem step by step here, mostly following the level-set method presented in [2].

1. **Construction of level-set function.** We start by constructing an initial level-set function, which gives the distance of an arbitrary $x \in \Omega = \mathbb{R}$ to the interface $s(t)$, which has initial position $s(t^0) = s_0$. Therefore, the level-set function is initialized as follows.

$$\phi(x,0) = \begin{cases} |x - s_0| & \text{if } x < s_0 \quad (4.4a) \\ 0 & \text{if } x = s_0 \quad (4.4b) \\ -|x - s_0| & \text{if } x > s_0 \quad (4.4c) \end{cases}$$

This function is called the level-set function, since it assigns a value of 0 to x if $x = s(t^0)$ and otherwise a non-zero value. We want to update this function in such a way that this function equals 0 whenever $x = s(t)$, so $\phi(s(t), t) = 0 \forall t \geq t^0$. So, the zero level-set of the function $\phi(x,t)$ is the interface $s(t)$. The idea of the level-set method is to move the position of the level-set with a certain time-dependent speed $v(t)$ and to update the values of the temperature $T(x,t)$ using the value of the zero level-set. In this way, it is not necessary to explicitly track the front, which makes it easier to apply this method to more-dimensional problems.

2. **Updating the level-set function** Using the Stefan condition and the temperature distribution $T(x,t)$ at time t, we can compute the velocity with which the front moves:

$$v(t) = \frac{ds}{dt} = T_x|_{x \downarrow s(t)} - T_x|_{x \uparrow s(t)}, \qquad x \in \Gamma(t), \text{ Stefan condition} \quad (4.5)$$

However, this condition only provides us with the velocity at the moving boundary. We want to update the level-set function on the whole domain. Therefore, we need to extend the found front velocity $v(t)$ from Γ to Ω. This is done by finding a continuous extension $F(x,t)$ of the front velocity. This continuous extension can for example be found by solving

$$\frac{\partial F}{\partial \tau}(x,\tau) + S(\phi(x,t)\frac{\partial \phi}{\partial x}(x,t))\frac{\partial F}{\partial x}(x,\tau) = 0 \quad (4.6)$$

where S is the sign function and τ is some time step not necessarily related to the main time step(see [8]). Another option which only works in one-dimensional problems is taking $F(x,t) = v(t)$: we simply set the velocity extension on the whole domain Ω equal to the velocity at the front. Note that both velocity extensions satisfy the advection equation:

$$\frac{\partial \phi}{\partial t} + F(x,t)\frac{\partial \phi}{\partial x} = 0 \quad (4.7)$$

By using the fact that we can compute $\frac{\partial \phi}{\partial x}$ since we know what $\phi(x,t)$ is at each time step and substituting the extended velocity $F(x,t)$, we can update the level-set function by solving this equation.

3. **Re-initialization of the level-set function.** Using continuous extensions, it can happen that the signed distance function is no longer a distance function. In this case, the function has to be re-initialized such that $\phi(s(t),t) = 0$ still holds and the spatial derivative of the function equals $\frac{\partial \phi}{\partial x} = -1$ for all $x \in \Omega$. Fortunately, we can relatively easy compute an exact signed distance function fulfilling these two conditions by iterating the following equation, with S denoting the signed distance function and τ a fictitious time step,

$$\frac{\partial \phi}{\partial \tau}(x,\tau) = S(\phi(x,t))(1 - \left|\frac{\partial \phi}{\partial x}(x,\tau)\right|) \quad (4.8)$$

until steady-state is reached. This steady-state function fulfills both of our conditions: it is a signed-distance function with negative slope -1 and it crosses the x-axis at position $x = s(t)$.

4. **Updating the temperature.** The last important step in the level-set method is to update the values of the temperature $T(x,t)$. We perform this step after moving the signed-distance function with extended velocity $F(x,t)$ and re-initializing it to an exact signed distance

function. In order to update the temperature, we basically need to solve the heat equations over the whole domain Ω. We can use the re-initialized signed distance function to find out if a point x is close to the interface $s(t)$. If this is the case, we use interpolation to approximate the double derivatives T_{xx} of the temperature. For points far away from the moving interface, we solve the heat equation using a standard numerical method (finite difference or finite volume schemes with central differences).

The described level-set method can be summarized in the following steps.

1. Set the initial values for the temperature $T(x,t)$ and define the signed distance function $\phi(x,t)$.
2. Compute the extended velocity field $F(x,t)$, which is the continuous extension of the velocity $v(t) = s_t$, which can be computed using the Stefan condition.
3. Update the distance function by using the function $\phi_t + F\phi_x = 0$. The zero level-set of the equation ϕ gives the new interface position.
4. Re-initialize ϕ in order to make it an exact signed distance function again.
5. Find the temperature field $T(x,t,)$ by discretizing the heat equation. For points far from the interface, solve the heat equation by using an implicit centered differences numerical method. For points close to the interface, approximate T by interpolating from the points at one side of the surface (with the same phase), using ϕ.
6. Repeat steps 2 through 5 to update the values of ϕ and T.

Note that we only described the level-set method in terms of exact partial differential equations. We did not yet choose a discretization of time and space. The discretization of the described level-set method is introduced in Chapter 7. Before we describe the discretization, we will first consider several numerical schemes applied to the advection equation and the heat equation, in order to be able to choose the most suitable numerical scheme to solve the heat and the advection sub-problems in the Stefan problem.

5 Choosing numerical schemes for developing a level-set method

The main goal of this work is to reach second order accuracy in solving the Stefan problem with the level-set method. In order to reach this accuracy, we need to choose numerical schemes which are sufficiently accurate. In this section, we investigate several numerical schemes to solve the advection problem and the heat problem. In the first part of this chapter, 5.1, we consider the heat problem, which is important when updating the temperature in the Stefan problem. In the second and last section 5.2 of this chapter, we consider numerical schemes for solving the advection problem. This problem has to be solved numerically if we want to describe the evolution of the moving boundary in the Stefan problem.

5.1 Choosing a numerical scheme for solving the heat problem

First of all, we want to discretize the temperature update procedure in the level-set method in such a way that we obtain second order accuracy as well in space as in time. In this chapter we will only deal with a heat problem with fixed boundaries - in contrary to the Stefan problem, where the moving boundary is a characteristic part of the problem - in order to be able to make a good comparison between several methods for solving the heat problem. We compare three numerical schemes which are often used to solve the heat problem: the FTCS scheme (Forward Time, Centered Space), the BTCS-scheme (Backward Time, Centered Space) and the Crank-Nicolson method.

In order to make a fair comparison between these three schemes, we use a one-phase heat problem of which the analytical solution is known. Note that we do not use the Stefan problem here, since we have to take the moving boundary into consideration then, which largely complicates the problem and makes it more difficult to compare the different numerical methods in their accuracy. Therefore, we consider a one-dimensional box parallel to the x-axis of length L with its left side at $x = 0$. The box is filled with a certain material with density ρ, thermal conductivity k and specific heat capacity c. The material has an initial temperature distribution $T(0, t)$ and in the whole domain, the material is in the same phase, so we do not have to deal with phase changes. This situation is sketched in the following picture.

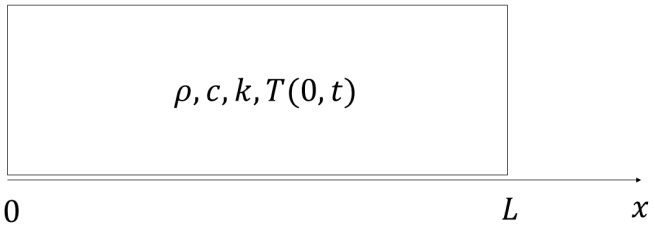


Figure 4: Graphical representation of the represented problem

For the boundaries $x = 0, x = L$, we consider in the homogeneous Dirichlet boundary conditions: $T(0, t) = T(L, t) = 0$. In physics, this corresponds to a system where the temperature at the boundary is kept fixed at $T = 0$.

The thing we are interested in and which we want to find is the distribution of the temperature in the rod after a certain time $t > 0$, so we want to find $T(x, t)$, given the initial condition $T(x, 0)$ and the boundary conditions. We simplify the problem by choosing $\rho = k = c = L = 1$ which leads to the following problem to solve:

$$\begin{aligned} \text{PDE:} \quad & \frac{\partial T}{\partial x} = \frac{\partial^2 T}{\partial x^2} && 0 < x < 1 \\ \text{IC:} \quad & T(x,0) = f(x) && 0 \leq x \leq 1 \\ \text{BC (Dirichlet):} \quad & T(0,t) = T(1,t) = 0 && t > 0 \end{aligned}$$

An analytical solution to this described problem is given in [1] by

$$T(x,t) = \sum_{n=1}^{\infty} B_n \sin(n\pi x) \exp\left(-n^2\pi^2 t\right) \tag{5.1}$$

where the B_n are given by

$$B_n = 2 \int_0^1 \sin(n\pi x) f(x) dx. \tag{5.2}$$

In this work, we will furthermore assume for the sake of simplicity that the initial temperature is given by 1 on the whole domain: $f(x) = T(x,0) = \sin(\pi x)$. This assumption allows us to simplify the previous expressions by using orthogonality of sinuses to the following:

$$T(x,t) = \sin(\pi x) \exp\left(-\pi^2 t\right). \tag{5.3}$$

A plot of this analytical solution is shown in Figure 5. We see that in the beginning, the temperature is, as we initialized it, a sine function. As we expect, after a long time the temperature profile is approximately equal to zero.

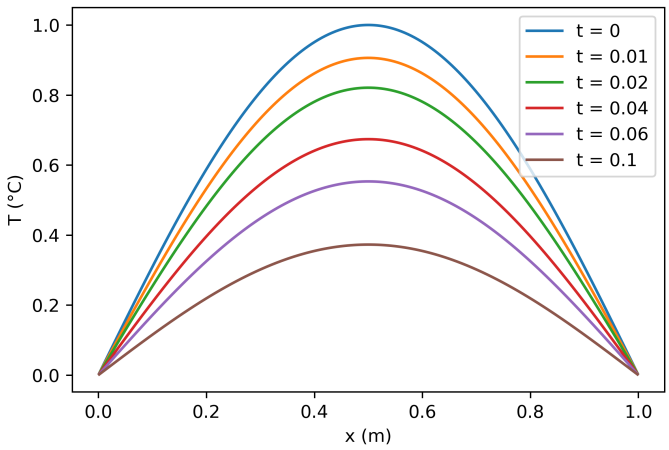


Figure 5: Analytical solution of the homogeneous Dirichlet problem for different times t, with $t \in \{0.0, 0.01, 0.02, 0.04, 0.05, 0.1\}$

Now that we have defined the problem we want to solve numerically, we discretize our spatial domain $\Omega = [0,1]$ with a uniform spatial grid into M grid points, i.e. $\Delta x = 1/M, x_i = (i - 1/2)\Delta x, i = 1, \ldots, M$. We also choose a temporal uniform discretization with time step Δt, so $t^n = n\Delta t, n = 0, \ldots, N$. We will now consider the mentioned numerical schemes. First we look at the FTCS-scheme, then we consider the BTCS-scheme and finally we present the Crank-Nicolson scheme. We start our models at $t = 0$ and we compare the numerical error of these schemes at time $t^N = 0.1$. Recall that the numerical L^2-error of the temperature profile at time $t = t^N$ is given by

$$||e^n||_2 = (\Delta x \sum_{i=1}^{M} |e_i^n|^2)^{1/2} \tag{5.4}$$

with $e_i^n = T(x_i, t^n) - T_i^n$.

5.1.1 FCTS-scheme

The first method which is considered is the Forward Euler method (for time discretization) combined with the central difference method (for spatial discretization). This method is theoretically first order accurate in time and second order accurate in space. The discretized (simplified) for points in the domain heat equation given by

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2} \tag{5.5}$$

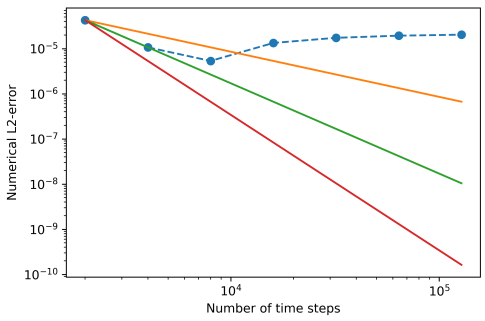
Note that this method is explicit. This expression describes a numerical scheme in the inner points of the domain. However, we have to find a way to implement the boundary conditions at $x = 0$ and $x = 1$. At the boundary $x = 0$, we know that $T = 0$ for all n since we have to deal with homogeneous Dirichlet boundary conditions. Therefore, we can write the following condition for the average of the virtual point T_0^n and T_n^1 :

$$T(0,t) \approx \frac{T_0^n + T_1^n}{2} + \mathcal{O}(\Delta x^2) = 0 \tag{5.6}$$

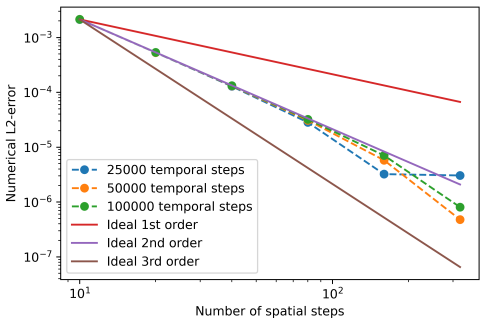
which leads to the condition $T_0^n = -T_1^n$. In a similar way, we can derive for the other boundary that $T_{M+1}^n = -T_M^n$. Using this boundary conditions and the expression for the other points, we see that we can evaluate the temperature at time step $t = t^{n+1}$ by solving the following matrix equation. Note that this method is explicit and that we have to satisfy the stability condition $\frac{\Delta t}{\Delta x^2} \leq 1/2$ in order to get convergent, stable solutions.

$$\begin{pmatrix} T_1^{n+1} \\ T_2^{n+1} \\ \vdots \\ T_{M-1}^{n+1} \\ T_M^{n+1} \end{pmatrix} = \begin{pmatrix} 1 - 3\frac{\Delta t}{\Delta x^2} & \frac{\Delta t}{\Delta x^2} & 0 & 0 & 0 \\ \frac{\Delta t}{\Delta x^2} & 1 - 2\frac{\Delta t}{\Delta x^2} & \frac{\Delta t}{\Delta x^2} & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & \frac{\Delta t}{\Delta x^2} & 1 - 2\frac{\Delta t}{\Delta x^2} & \frac{\Delta t}{\Delta x^2} \\ 0 & 0 & 0 & \frac{\Delta t}{\Delta x^2} & 1 - 3\frac{\Delta t}{\Delta x^2} \end{pmatrix} \begin{pmatrix} T_1^n \\ T_2^n \\ \vdots \\ T_{M-1}^n \\ T_M^n \end{pmatrix} \tag{5.7}$$

Having computed the numerical scheme to solve the heat equation for the described problem, we can fix the number of spatial steps at some number and vary the number of time steps to observe how the numerical L^2-error behaves as a function of time step size. This is shown in Figure 7a. We can also fix the number of time steps and vary the number of spatial steps to observe the numerical L^2-error as the number of spatial step size. This is shown in Figure 7b.



(a) Numerical error at time $t = 0.1$ for fixed number of spatial steps $M = 100$ and varying time step size.



(b) Numerical error at time $t = 0.1$ for the FTCS-scheme for fixed number of time steps $N = 25000, 50000$ and $N = 100000$ and a varying spatial step size.

Figure 6: Numerical error of the numerical solution to the homogeneous Dirichlet problem for the FTCS-scheme as a function of the spatial step size and as a function of the temporal step size.

We see in Figure 7b that the method is clearly second order accurate in space. The blue line for $M = 25000$ becomes constant after $M = 160$ spatial steps. This can be explained by the fact that the numerical error due to the fixed number of time steps becomes the dominant error here, and since this error is fixed, the numerical error becomes constant. In Figure 7a, we see that

the numerical error as a function of the number of time steps becomes approximately constant for $N \geq 16000$. This constant numerical error is probably the fixed numerical error due to fixed spatial step size. It would be interesting to consider larger numbers of spatial steps than $M = 100$. However, note that the stability condition $\Delta t/\Delta x^2 \leq 1/2$ in that case requires that the number of time step sizes will increase quadratic. This will make that the error due to time step size will decrease faster then the error due to spatial step size, which will make it even more difficult to compare the numerical error due to spatial step size to the numerical error due to time step size. This effect makes it impossible to investigate the numerical error due to time step size separately from the spatial step size. Therefore, we have to draw our conclusions on the small segment between $N = 2000$ and $N = 8000$. We see that the numerical error converges here with first order accuracy.

Having considered the FCTS-method, we can conclude that this method is indeed first order accurate in time and second order accurate in space. Advantage of this method is that it is explicit (which means that the computational time is low compared to implicit methods). On the other hand, the fact that it is an explicit method makes that we have to satisfy the stability condition $\Delta t/\Delta x^2 \leq 1/2$, which makes that we have to choose our time step very small in order to obtain stable solutions. Furthermore, if we want to develop a second order method for the Stefan problem, it does not seem to be handy to use first-order accurate methods.

5.1.2 BTCS-scheme

Next, we consider the BTCS-scheme as solution for the Stefan problem, a method which is first order accurate in time and second order accurate in space. A big advantage of this method compared to the Forward Euler method is that this method is unconditionally stable. However, since it is an implicit method, this method takes more computing power.

The discretized heat equation for the Backward Euler Method is given by

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{\Delta x^2} \tag{5.8}$$

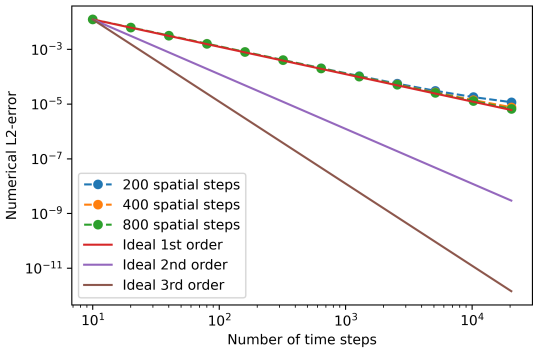
The boundary conditions for the Dirichlet problem imply that $T_1^n = -T_0^n$ (we do not show the derivation of this equation, since it is very similar to the derivation of the boundary conditions for the FTCS-scheme). Therefore, the matrix equation we have to solve in order to find the temperature for the following time step is given by

$$\begin{pmatrix} T_1^n \\ T_2^n \\ \vdots \\ T_{M-1}^n \\ T_M^n \end{pmatrix} = \begin{pmatrix} 1+3\frac{\Delta t}{\Delta x^2} & -\frac{\Delta t}{\Delta x^2} & 0 & 0 & 0 \\ -\frac{\Delta t}{\Delta x^2} & 1+2\frac{\Delta t}{\Delta x^2} & -\frac{\Delta t}{\Delta x^2} & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & -\frac{\Delta t}{\Delta x^2} & 1+2\frac{\Delta t}{\Delta x^2} & -\frac{\Delta t}{\Delta x^2} \\ 0 & 0 & 0 & -\frac{\Delta t}{\Delta x^2} & 1+3\frac{\Delta t}{\Delta x^2} \end{pmatrix} \begin{pmatrix} T_1^{n+1} \\ T_2^{n+1} \\ \vdots \\ T_{M-1}^{n+1} \\ T_M^{+1} n \end{pmatrix} \tag{5.9}$$

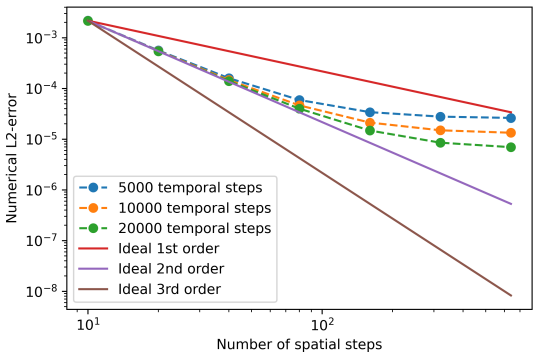
The numerical error (in the L^2-norm) at time $t = 0.1$ for this BCTS-method is shown in Figure 7, where in 7a the spatial step size is fixed and the temporal step size varied. In Figure 7b, the time step size is fixed and the spatial step size is varied.

In Figure 7, we see in the first place that the BTCS-scheme appears to be first order accurate in time: it follows the ideal first order almost perfectly for the investigated range of time steps. Furthermore, we see that it also appears to be second order accurate in space. However, since the numerical error due to time step size is relatively big compared to the spatial numerical error, we see that the numerical error as a function of the number of spatial steps becomes constant - it attains the value of the fixed numerical error due to time step size.

So, concluding about this method, we saw that the BTCS-scheme is second order accurate in space and first order accurate in time. Furthermore, a big advantage of this method is that we do not have to satisfy stability conditons like we had to for the FTCS-scheme. This compensates for the extra time necessary to solve matrix equations (instead of simply solving matrix multiplications, what we had to do to solve the FTCS-scheme). However, since the scheme is only first order accurate in time, it does not seem to be the best starting choice for our level-set method if we want to reach second order accuracy.



(a) Numerical error at time $t = 0.1$ for the BTCS-scheme for fixed spatial step $M = 200, 400$ and $M = 800$ and a varying time step size. Note that the blue and orange line are almost completely covered by the green line.



(b) Numerical error at time $t = 0.1$ for the BTCS-scheme for fixed number of time steps $N = 5000, 10000$ and $N = 20000$ and a varying spatial step size.

Figure 7: Numerical error of the numerical solution to the homogeneous Dirichlet problem for the BTCS-scheme as a function of the spatial step size and as a function of the temporal step size.

5.1.3 Crank-Nicolson scheme

The last scheme used to solve the heat equation considered in this report is the Crank-Nicolson scheme. This scheme is a so-called semi-implicit scheme and it is second order accurate in as well space as time. It is, just as the BTCS-scheme, an implicit method and it is therefore unconditionally stable. The discretized heat equation for this method is given by

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \frac{1}{2\Delta x^2}((T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}) + (T_{i+1}^n - 2T_i^n + T_{i-1}^n)) \tag{5.10}$$

We can solve this equation for new temperatures in terms of old temperatures as follows.

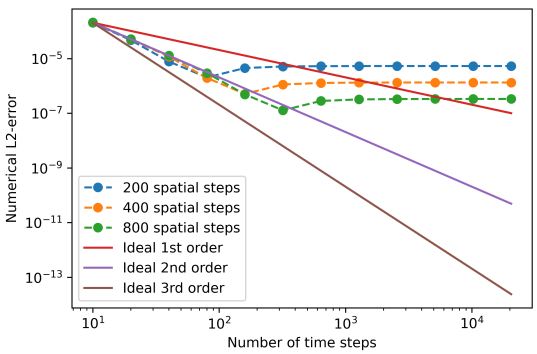
$$-\alpha T_{i-1}^{n+1} + (2+2\alpha)T_i^{n+1} - \alpha T_{i-1}^{n+1} = \alpha T_{i-1}^n + (2-2\alpha)T_i^n + \alpha T_{i+1}^n \tag{5.11}$$

where $\alpha = \frac{\Delta t}{\Delta x^2}$. Applying the Dirichlet boundary conditions, which is not shown since this derivation is done in a similar way for the FTCS- and BTCS-scheme, we find the following matrix equation.

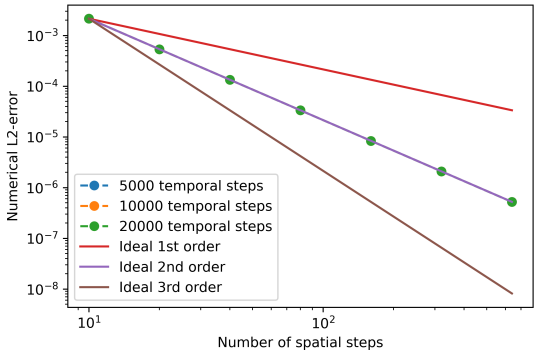
$$\begin{pmatrix} 2+3\alpha & -\alpha & 0 & 0 & 0 \\ -\alpha & 2+2\alpha & -\alpha & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & -\alpha & 2+2\alpha & -\alpha \\ 0 & 0 & 0 & -\alpha & 2+3\alpha \end{pmatrix} \begin{pmatrix} T_0^{n+1} \\ T_1^{n+1} \\ \vdots \\ T_{M-1}^{n+1} \\ T_M^{n+1} \end{pmatrix} = \begin{pmatrix} \alpha(2-3\alpha)T_1^n + \alpha T_2^n \\ \alpha T_1^n + (2-2\alpha)T_2^n + \alpha T_3^n \\ \vdots \\ \alpha T_{M-2}^n + (2-2\alpha)T_{M-1}^n + \alpha T_M^n \\ \alpha T_{M-1}^n + (2-3\alpha)T_M^n \end{pmatrix} \tag{5.12}$$

Note that the vector on the right hand side is known, as well as the $M \times M$-matrix on the left hand side. Therefore, we need to solve this matrix equation in order to find the temperature at the next time step. This method gives also very accurate results for small time and space step sizes. In Figure 8 , the numerical accuracy of the CN-scheme (Crank-Nicolson scheme) is presented.

In Figure 8, we see that this method seems to be second order accurate. For as well space as time, the numerical error follows the ideal second order line (until it reaches a constant numerical error, which is probably caused by the fixed time step size). This confirms the theory that this method is of second order accuracy for as well space as time. In the continuation of this report, we will therefore use the Crank Nicolson scheme to solve the heat equation, since this method gives a good, second order accurate base for solving the temperature update part of the Stefan problem.



(a) Numerical error at time $t = 0.1$ for the CN-scheme for fixed spatial step $M = 200, 400$ and $M = 800$ and a varying time step size. Note that the blue and orange line are almost completely covered by the green line.



(b) Numerical error at time $t = 0.1$ for the CN-scheme for fixed number of time steps $N = 5000, 10000$ and $N = 20000$ and a varying spatial step size.

Figure 8: Numerical error of the numerical solution to the homogeneous Dirichlet problem for the CN-scheme as a function of the spatial step size and as a function of the temporal step size.

5.2 Choosing a numerical scheme for solving the advection problem

Now that we have established which method to use for solving the heat equation, we want to consider several methods for solving the advection equation and to compare these methods in their accuracy. In this section, we will respectively consider the upwind method, the Leapfrog method and the Lax-Wendroff method.

We compare them for their accuracy in solving the following differential equation

$$\frac{\partial u}{\partial t} + v(x,t)\frac{\partial u}{\partial x} = 0. \tag{5.13}$$

This is the general form of a one-dimensional advection equation: the equation that governs the motion of a scalar field, in the case of the level-set method we use it to update the signed distance function. In this section however, we will test the three methods on another numerical problem, which is given below.

In order to solve the problem, we need to choose a certain initial condition $u(x,0) = u_0(x)$. The analytical solution to the problem is given then by $u(x,t) = u_0(x-vt)$ [7]. In this report, we will consider the initial condition given by

$$u_0(x) = \begin{cases} 0 & 0 \leq x \leq 1 \quad (5.14a) \\ \sin^4(\pi\dfrac{x-1}{x-5}) & 1 < x < 5 \quad (5.14b) \\ 0 & 5 \leq x \leq 10 \quad (5.14c) \end{cases}$$

on the domain $\Omega = [0, 10]$ with constant velocity $v(x,t) = 1$. As boundary conditions we choose the periodic boundary conditions. In Figure 9, the analytical solution is shown at $t = 4.0$. Note that the solution to the advection equation is simply given by the initial condition shifted to the right with a distance vt: we have $u(x,t) = u_0(x-vt)$. In this section, we will compare the solutions obtained by the first order Upwind method, Lax-Wendroff method and the Leapfrog method with the analytical solution at time $t^N = 4.0$. We discretize the domain with a uniform spatial grid into M grid points, with $\Delta x = 1/(M-1)$ and $x_i = (i-1)\Delta x, i = 1, \ldots, M$. The numerical error at $t = t^N$ time is given by

$$||e^n||_2 = (\Delta x \sum_{i=1}^{M} |e_i^n|^2)^{1/2} \tag{5.15}$$

with $e_i^n = T(x_i, t^n) - T_i^n$.

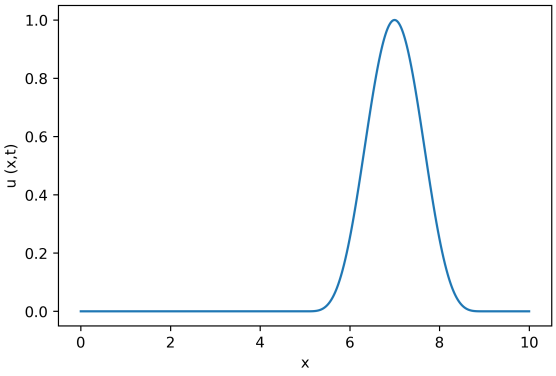


Figure 9: Analytical solution of the advection problem with periodic boundary conditions at time $t = 4.0$.

5.2.1 Upwind method

The first method considered is the first order upwind method. This method is first order accurate in time and also first order accurate in space and it is an explicit scheme. This scheme is given by

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \begin{cases} -v\dfrac{u_i^n - u_{i-1}^n}{\Delta x} & \text{if } v_i^n > 0 \quad (5.16a) \\ -v\dfrac{u_{i+1}^n - u_i^n}{\Delta x} & \text{if } v_i^n < 0 \quad (5.16b) \end{cases}$$

where v is assumed to be constant (as is the case in our problem). In other words: if the solution propagates to the left, the values on the right hand side of the solution are used to compute its propagation and vice versa for propagation to the right. Introducing the definitions $a_{max} = \max(a, 0)\frac{\Delta t}{\Delta x}$ and $a_{min} = \min(a, 0)\frac{\Delta t}{\Delta x}$, we can write

$$u_i^{n+1} = a_{max}u_{i-1}^n + (1 + a_{min} - a_{max})u_i^n - a_{min}u_{i+1}^n \tag{5.17}$$

Using periodic boundary conditions, we know that $u_1^n = u_{M+1}^n$, so $u_1^n = a_{max}u_M^n + (1 + a_{min} - a_{max})u_1^n - a_{min}u_2^n$ and we can find a similar expression for the other boundary. This leads to the following matrix.

$$\begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_N^{n+1} - 1 \\ u_N^{n+1} \end{pmatrix} = \begin{pmatrix} 1 + a_{min} - a_{max} & -a_{min} & 0 & \cdots & a_{max} \\ a_{max} & 1 + a_{min} - a_{max} & -a_{min} & \cdots & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & \ddots & \ddots & 0 \\ 0 & 0 & a_{max} & 1 + a_{min} - a_{max} & a_{min} \\ -a_{min} & 0 & 0 & a_{max} & 1 + a_{min} - a_{max} \end{pmatrix} \begin{pmatrix} u_1^n \\ u_2^n \\ \vdots \\ u_N^n - 1 \\ u_N^n \end{pmatrix} \tag{5.18}$$

This scheme is stable if the following CFL-condition is satisfied.

$$\frac{v\Delta t}{\Delta x} \leq 1 \tag{5.19}$$

Having derived the system of equations which should be solved, we can compute the numerical accuracy of this method. To find the numerical accuracy, we choose the temporal step size as a linear function of the spatial step size: $\Delta t = a\Delta x$ (or, in other words: we keep the CFL-number fixed at a). If the scheme is first order accurate in as well space and time, we see that the numerical error is of order

$$\mathcal{O}(\Delta x) + \mathcal{O}(\Delta t) = \mathcal{O}(\Delta x) + \mathcal{O}(a\Delta x) = \mathcal{O}(\Delta x) \tag{5.20}$$

so by only showing the numerical error as a function of the number of spatial steps, we would be able to see if this method is indeed first order accurate in space and time. In Figure 10, the numerical error is shown as a function of the number of spatial steps, where the time step size is chosen as $\Delta t = a\Delta x$, with $a \in \{1/4, 1/2\}$.

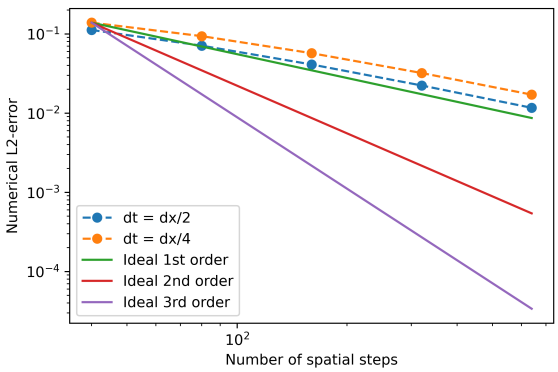
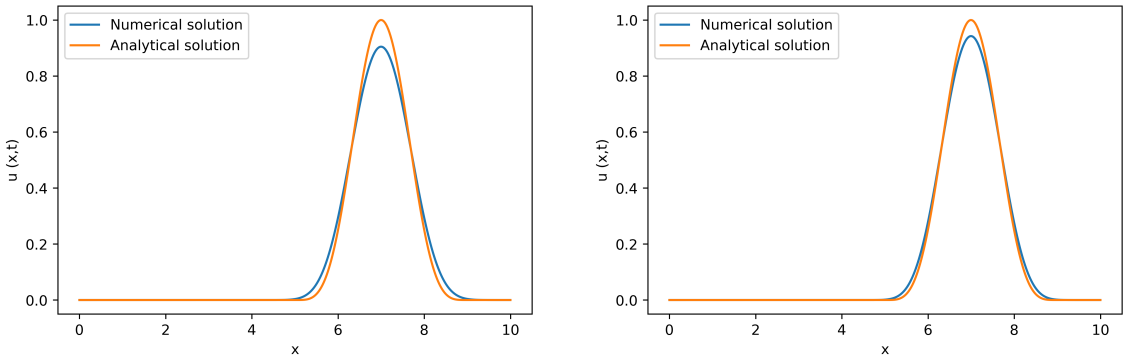


Figure 10: Numerical error at $t = 4.0$ for the Upwind Method. The time step is chosen as $\Delta t = a\Delta x$, with $a \in \{1/2, 1/4\}$, as shown in the legend.

In Figure 10, we see that this method is indeed (almost) first order accurate in space and time. What is also good to note is that this method can lead to numerical diffusion (a mismatch in phase between the analytical and numerical solution) and numerical dispersion (damping of the amplitude) when $a \neq 1$. This last effect is clearly visible in Figure 11, where the numerical and the analytical solutions are shown for $a = 1/8$ and $a = 1/2$.



(a) $a = 1/8, M = 400$ (b) $a = 1/2, M = 400$

Figure 11: Numerical and analytical solutions to the periodic boundary conditions advection problem for $a = 1/2$ and $a = 1/8$, both for $M = 400$ spatial steps.

In Figure 11, we see that the numerical dispersion decreases when we increase a. For $a = 1$, the solution to the upwind method equals the analytical solution, so there will be no dispersion at all for $a = 1$. To conclude, we see that the first order Upwind method is first order accurate in space and time. Furthermore, this method shows strong effects of numerical dispersion and numerical diffusion for low values of a, which makes it less useful for solving the advection problem.

5.2.2 Leapfrog method

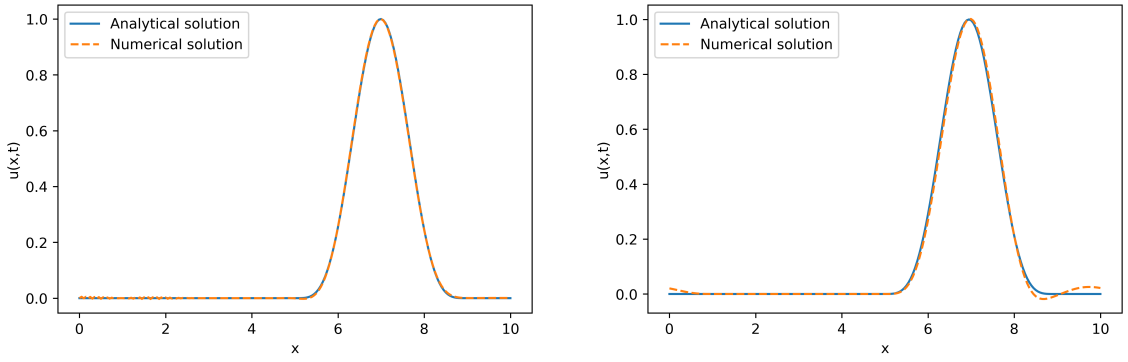
The next method we investigate is the Leapfrog method. This is a method which is second order accurate in as well space as time and therefore promising for this report. The Leapfrog method uses the values at the previous two timesteps to estimate the value of u at the next time step: it is a so-called multi-step method. This can be expressed in the following equation.

$$u_j^{n+1} = u_j^{n-1} - \alpha(u_{j+1}^n - u_{j-1}^n) \tag{5.21}$$

where $\alpha = \frac{v\Delta t}{\Delta x}$. This method is stable if $\alpha \leq 1$. Note that this scheme is a two-level-scheme, so we need the values of the timesteps n and $n-1$ to get the value of $n+1$. A big advantage of the Leapfrog method is that there is theoretically no amplitude dissipation. The matrix equation using the given equation can be expressed as follows.

$$\begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{M-1}^{n+1} \\ u_M^{n+1} \end{pmatrix} = \begin{pmatrix} u_1^{n-1} \\ u_2^{n-1} \\ \vdots \\ u_{M-1}^{n-1} \\ u_M^{n-1} \end{pmatrix} + \begin{pmatrix} 0 & -\alpha & 0 & \cdots & 0 & 0 & \alpha \\ \alpha & 0 & -\alpha & \cdots & 0 & 0 & 0 \\ & & & \ddots & & & \\ 0 & 0 & 0 & \cdots & \alpha & 0 & -\alpha \\ \alpha & 0 & 0 & \cdots 0 & -\alpha & 0 & \end{pmatrix} \begin{pmatrix} u_1^n \\ u_2^n \\ \vdots \\ u_{M-1}^n \\ u_M^n \end{pmatrix} \tag{5.22}$$

Again, we investigate the numerical accuracy of this method by plotting the error as a function of the spatial step size, where the time step size is given by $\Delta t = a\Delta x$ for different values of a. The Leapfrog does not give the analytical result for $a = 1$ (as we saw for the upwind method). On the contrary, it becomes more accurate for smaller CFL-numbers. This can be observed in Figure 12. For $a = 1$, we see a sort of little "dip" after the peak and around $x = 0$, while for $a = 1/8$ the function is much smoother.



(a) $a = 1/8, M = 200$ (b) $a = 1, M = 200$

Figure 12: Numerical and analytical solutions to the periodic boundary conditions advection problem for $a = 1$ and $a = 1/8$, both for $M = 200$ spatial steps.

Now, we show the numerical error as a function of the spatial step size in Figure 13. We see that the numerical error is initially of second order for all values of a. However, after a certain number of spatial steps, it seems to become first order accurate. For small values of a, for example $a = 1/128$, we see that the method is second order accurate for the whole investigated range of spatial steps. So, a big disadvantage of the Leapfrog method is that the ratio between temporal and spatial step size has to be small in order to get second order accurate results, which leads to large calculation times.

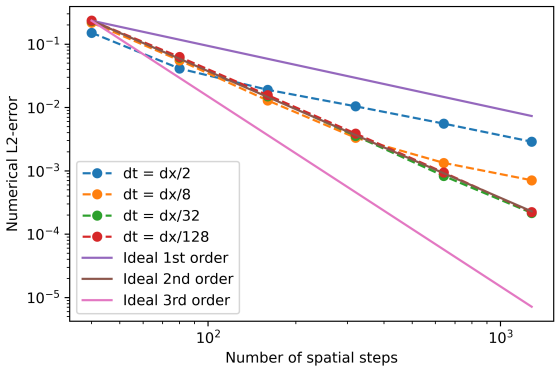


Figure 13: Numerical error at $t = 5.0$ for the Leapfrog Method. The time step is chosen as $\Delta t = a\Delta x$, with $a \in \{1/2, 1/8, 1/32, 1/128\}$, as shown in the legend.

5.2.3 Lax-Wendroff method

The third and last method for solving the advection equation considered in this report is the Lax-Wendroff method, which is second order accurate in space and time. This method is a so-called multi-step method. First, we approximate the value of the function $u(x,t)$ at half time steps. After this, we use the central difference method to approximate the spatial derivative using central differences. This leads to the following equations.

$$u_{i-1/2}^{n+1/2} = \frac{1}{2}(u_i^n + u_{i-1}^n) + \frac{v\Delta t}{2\Delta x}(u_i^n - u_{i-1}^n) \tag{5.23}$$

$$u_{i+1/2}^{n+1/2} = \frac{1}{2}(u_i^n + u_{i+1}^n) + \frac{v\Delta t}{2\Delta x}(u_{i+1}^n - u_i^n) \tag{5.24}$$

$$u_i^{n+1} = u_i^n - \frac{v\Delta t}{\Delta x}(u_{i+1/2}^{n+1/2} - u_{i-1/2}^{n+1/2}) \tag{5.25}$$

Using the periodic boundary conditions, we can write this system as the following matrix equation.

$$\begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{M-1}^{n+1} \\ u_M^{n+1} \end{pmatrix} = \begin{pmatrix} 1-\alpha^2 & \frac{\alpha}{2}(\alpha+1) & 0 & 0 & \frac{\alpha}{2}(\alpha-1) \\ \frac{\alpha}{2}(\alpha-1) & 1-\alpha^2 & \frac{\alpha}{2}(\alpha+1) & 0 & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & 0 & \frac{\alpha}{2}(\alpha-1) & 1-\alpha^2 & \frac{\alpha}{2}(\alpha+1) \\ \frac{\alpha}{2}(\alpha+1) & 0 & 0 & \frac{\alpha}{2}(\alpha-1) & 1-\alpha^2 \end{pmatrix} \begin{pmatrix} u_1^n \\ u_2^n \\ \vdots \\ u_{M-1}^n \\ u_M^n \end{pmatrix} \tag{5.26}$$

where $\alpha = \frac{v\Delta t}{\Delta x}$. This method is stable if $\alpha \leq 1$ (or: $v \leq \frac{\Delta x}{\Delta t}$. An advantage of this method with respect to the upwind method is that the diffusion and dispersion effects for this method are relatively small. Note that the numerical solution given by the Lax-Wendroff method is exactly equal to the analytical solution whenever $\alpha = 1$.

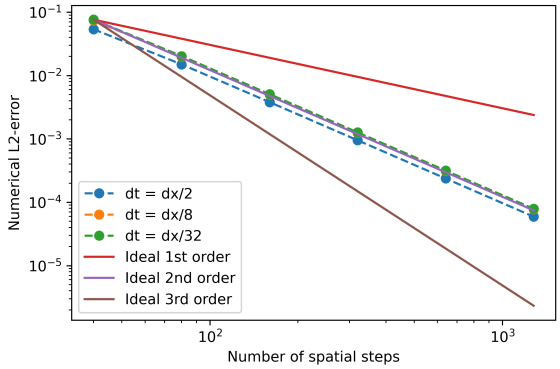


Figure 14: Numerical error at $t = 5.0$ for the Lax-Wendroff method. The time step is chosen as $\Delta t = a\Delta x$, with $a \in \{1/2, 1/8, 1/32\}$, as shown in the legend. The orange line ($\Delta t = \Delta x/8$) is completely covered by the green line ($\Delta t = \Delta x/32$)

.

We show the numerical error again as function of the spatial step size, with the temporal step size chosen as $\Delta t = a\Delta x$, for some different (positive) values of a in Figure 14. We see that this method is second order accurate in space and time, just as expected from the theory. Furthermore, an advantage of this method is that the accuracy of the method is not so dependent on the value of the ratio a as we saw for the Leapfrog method. For all investigated numbers a, we see that the Lax-Wendroff method gives second order accurate solutions. Therefore, we will choose this numerical method as point of departure for solving the advection problem while developing our level-set method.

6 Discretization

In the previous chapters, the Stefan problem was defined and the level-set method was introduced. In this chapter, we will apply this theory in order to develop a discretization of the level-set method for the Stefan problem. In section 6.1, the Stefan problem considered in this work is shortly formulated for the sake of clarity. In the following section, 6.2, a discretization of the spatial and temporal domain is given. Furthermore, an discretization of the temperature and the signed distance function is defined. After this, in section 6.3, the procedure the find the velocity of the moving boundary is described. Using this velocity, in section 6.4, we describe a discretization of the procedure to solve the advection problem for the moving boundary numerically. In section 6.5, we describe how the heat problem is discretized. Lastly, in section 6.6, it is explained how the numerical error in this work was computed.

6.1 Considered Stefan problem

This section briefly summarizes the Stefan problem considered in this work for the sake of clarity. We consider the Stefan problem with non-homogeneous Dirichlet boundary conditions on the domain $\Omega = [0, l]$. The Dirichlet boundary condition imposed to the temperature $T(x,t)$ is the (time-dependent) temperature of the similarity solution at boundaries. These boundary conditions are chosen in order to be able to make a good comparison between numerical and similarity solutions. We consider the Stefan problem with the following initial condition:

$$T(x,0) = \begin{cases} T_L > 0 & \text{if } x \in \Omega_L = [0, s(0)) \quad (6.1a) \\ 0 & \text{if } x = s(0) \quad (6.1b) \\ T_S < 0 & \text{if } x \in \Omega_S = (s(0), l] \quad (6.1c) \end{cases}$$

where T_L, T_S are constant and $s(0)$ denotes the position of the moving boundary at time $t = 0$. Using this initial condition and the boundary conditions, we can formulate the problem shortly as follows. We use the definitions $\alpha_L = \frac{k_L}{\rho_L c_L}$ and $\alpha_S = \frac{k_S}{\rho c_S}$.

The liquid region	$0 \leq x < s(t)$
$T_t = \alpha_L T_{xx}$,	Heat equation for the liquid region,
$T(0,t) = T_{ana,L}(t)$,	Boundary condition, $t \geq 0$
$T(x,0) = T_L > 0$,	Initial condition
The free boundary	$x = s(t)$
$\rho L s_t = k_S T_x\|_{x \downarrow s(t)} - k_L T_x\|_{x \uparrow s(t)}$	Stefan condition
$s(0) = s_0$,	Initial position of the melting interface
$T(s(t), t) = 0$,	Dirichlet condition at the interface
The solid region	$s(t) < x \leq l$
$T_t = \alpha_S T_{xx}$,	Heat equation for the solid region
$T(l,t) = T_{ana,S}(t)$,	Boundary condition, $t \geq 0$
$T(x,0) = T_S < 0$	For all $t, x \geq s(t)$

In these equations, $T_{ana,L}$ and $T_{ana,S}$ denote the values of the similarity solution at respectively $x = 0$ and $x = l$. Note that the similarity solution is known, so we do not have to make any estimates to find the values of this boundary condition. In the other part of this chapter, we will formulate the discretizations necessary to solve this problem numerically.

6.2 Space and time discretization and definitions

In this section, we discretize the equations which play a role in the Stefan problem. However, we first start with a discretization of time and space. In all of our computations, we take the domain

$\Omega = [0, l]$. We choose a uniform fixed grid with constant grid size Δx. Therefore, we set $\Delta x = 1/M$, where M is the total number of gridpoints on the grid. We also use a constant time step Δt and a total number of timesteps N. In the rest of the sections, we take the following definitions:

$$
\begin{aligned}
&x_i = (i - 1/2)\Delta x, && i = 1, \ldots, M \\
&K_i = [(i - 1/2)\Delta x, (i + 1/2)\Delta x], && i = 1, \ldots, M \\
&t^n = n\Delta t, && n = 0, \ldots, N \\
&s^n = s(t^n), && n = 0, \ldots, N \\
&\phi_i^n = \phi(x_i, t^n), && i = 1, \ldots, M,\ n = 0, \ldots, N \\
&T_i^n = T(x_i, t^n), && i = 1, \ldots, M,\ n = 0, \ldots, N \\
&\hat{T}_i^n = \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} T(x, t^n) dx, && i = 1, \ldots, M,\ n = 0, \ldots, N
\end{aligned}
$$

Furthermore, we define the grid point at the left side of the moving boundary as x_j. Note that the position of this grid point is a non-constant function of time. A schematic representation of the grid is given in Figure 15.

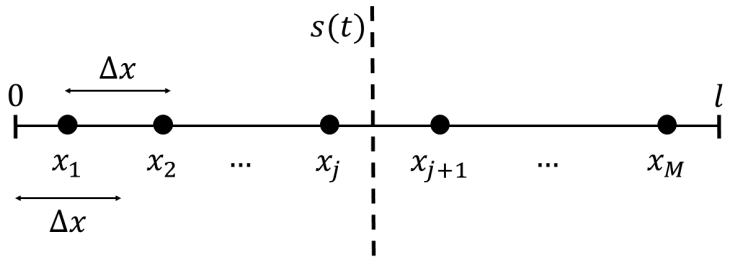


Figure 15: A graphical representation of the used discretized grid. The moving boundary position $s(t)$ is shown as a dashed line.

Note that the points x_1 and x_M do not coincide with the boundaries $x = 0$ and $x = l$: they are $\frac{1}{2}\Delta x$ apart from the boundaries.

6.3 Finding the front velocity and its extension

The first quantity we need to estimate, is the velocity of the moving boundary $v(t) = \frac{ds}{dt}$. This velocity is simply given by the Stefan condition:

$$\rho L \frac{ds}{dt} = k_S \frac{\partial T}{\partial x}|_{x \downarrow s(t)} - k_L \frac{\partial T}{\partial x}|_{x \uparrow s(t)}. \tag{6.2}$$

However, if we want to apply this condition, we need an estimate for the first derivatives of the temperature with respect to the spatial coordinate x. We estimate this derivative using one-sided second degree Lagrange polynomials. Before we give this polynomial, we introduce the coefficients r_1 and $_2$, which show the distance between the point x_j and the moving boundary and are given by the following definition:

$$r_1(t^n) = \frac{\phi_j^n}{\Delta x} = 1 - r_2(t^n), \tag{6.3}$$

where ϕ_j^n is the value of the signed distance function in the point $x = x_j$ at time $t = t^n$. Note that these coefficients can be computed numerically, using the numerical values of the signed distance function ϕ. In Figure 16 it is shown graphically what the coefficients r_1 and r_2 denote.

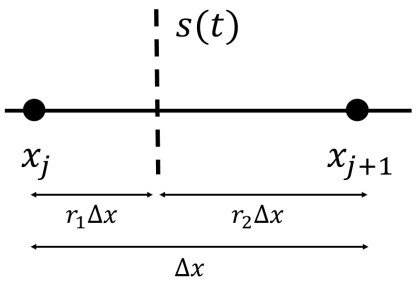


Figure 16: Graphical representation of the coefficients r_1 and r_2. Note that these coefficients multiplied with the spatial step size Δx are equal to the distance from the grid point x_j to the moving boundary $s(t)$. Also note that, since $s(t)$ is time-dependent, also x_j and the coefficients r_1, r_2 are time-dependent variables.

If the moving boundary is more than $\frac{1}{3}\Delta x$ away from the grid point x_j (i.e. $r_1 \geq 1/3$), we approximate the temperature around the boundary with the following Lagrange polynomial. Note that the third term of the approximating polynomial disappears because we assumed that $T_m = 0$.

$$\begin{aligned} P_L(x)(t^n) &= T_{j-1}^n \frac{(x - x_j)(x - s^n)}{(x_{j-1} - x_j)(x_{j-1} - s^n)} + T_j^n \frac{(x - x_{j-1})(x - s^n)}{(x_j - x_{j-1})(x_j - s^n)} \\ &= T_{j-1}^n \frac{(x - x_j)(x - s^n)}{(1 + r_1)\Delta x^2} - T_j^n \frac{(x - x_{j-1})(x - s^n)}{r_1 \Delta x^2} \end{aligned} \tag{6.4}$$

For the case that the moving boundary is closer than $\frac{1}{3}\Delta x$ to the moving boundary (i.e. $r_1 < 1/3$), we do not use the grid point x_j to construct the Lagrange polynomial, since the fact that the boundary and x_j are close to each other could lead to large inaccuracies in our Lagrange polynomial. Instead of the point x_j, we therefore use x_{j-2} in the construction of the polynomial, which leads to the following equation.

$$\begin{aligned} P_L(x)(t^n) &= T_{j-2}^n \frac{(x - x_{j-1})(x - s^n)}{(x_{j-2} - x_{j-1})(x_{j-2} - s^n)} + T_{j-1}^n \frac{(x - x_{j-2})(x - s^n)}{(x_{j-1} - x_{j-2})(x_{j-1} - s^n)} \\ &= T_{j-2}^n \frac{(x - x_{j-1})(x - s^n)}{(2 + r_1)\Delta x^2} - T_j^n \frac{(x - x_{j-2})(x - s^n)}{(1 + r_1)\Delta x^2} \end{aligned} \tag{6.5}$$

In Figure 17, we see the Lagrange polynomial for some specific Stefan problem. At the solid side, the Lagrange polynomial is computed in a similar way as for the liquid side. Note that this Lagrange polynomial is of second order, so it is part of a quadratic curve. However, since the interval at which we look is very small, it looks like a linear function.

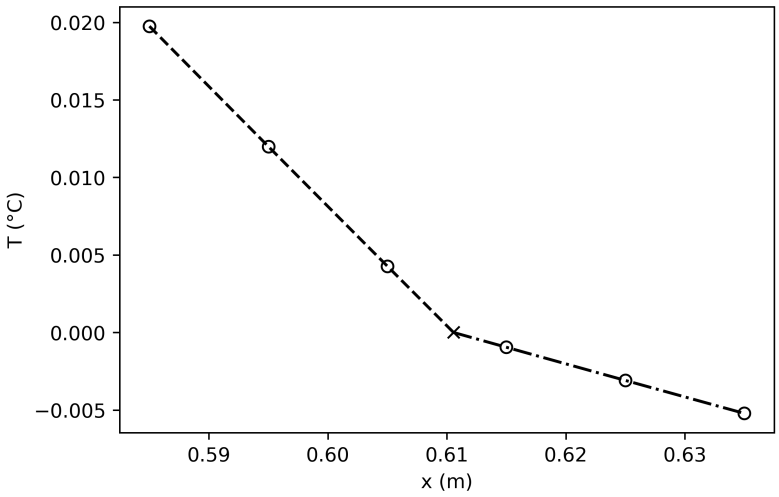


Figure 17: A graphical representation of the Lagrange polynomials around the moving boundary at time $t = t^n$. The boundary is positioned at $s^n \approx 0.61$ and is represented by the black cross point. The empty circles represent the numerically computed temperature values in the grid points $x_{j-2}, x_{j-1}, xj, x_{j+1}, xj + 2$ and x_{j+3} (from left to right). The dashed line shows the Lagrange polynomial on the liquid side, computed from the points x_{j-1}, x_j and s^n (since $r_1 > 1/3$ here). The dash-dotted line shows the Lagrange polynomial computed from the grid points s^n, x_{j+1}, x_{j+2} (since $r_2 > 1/3$).

If we take the first derivative with respect to x of the approximation Lagrange polynomials, we obtain the following estimate for the temperature gradient in the moving boundary from the liquid side for $r_1 \geq 1/3$:

$$P'_L(s^n)(t^n) = T^n_{j-1}\frac{s^n - x_j}{(1+r_1)\Delta x^2} - T^n_j\frac{s^n - x_{j-1}}{r_1\Delta x^2} \tag{6.6}$$

For $r_1 < 1/3$, we find:

$$P'_L(s^n)(t^n) = T^n_{j-2}\frac{s^n - x_{j-1}}{(2+r_1)\Delta x^2} - T^n_{j-1}\frac{s^n - x_{j-2}}{(1+r_1)\Delta x^2} \tag{6.7}$$

A similar derivation for the temperature gradient at the solid side of the moving boundary leads to the following approximation for $r_2 \geq 1/3$

$$P'_S(s^n)(t^n) = -T^n_{j+1}\frac{s^n - x_{j+2}}{r_2\Delta x^2} + T^n_{j+2}\frac{s^n - x_{j+1}}{(1+r_2)\Delta x^2} \tag{6.8}$$

and for $r_2 < 1/3$, we find

$$P'_S(s^n)(t^n) = -T^n_{j+2}\frac{s^n - x_{j+3}}{(1+r_2)\Delta x^2} + T^n_{j+3}\frac{s^n - x_{j+2}}{(2+r_2)\Delta x^2} \tag{6.9}$$

Using these approximations, we can use the Stefan condition. We estimate the moving boundary velocity v^n at time $t = t^n$ using the previous equations as follows:

$$v^n = \frac{1}{L}(k_L P'_S(s^n) - k_S P_L l(s^n)) \tag{6.10}$$

This velocity is the velocity at the moving boundary. To find a velocity extension to the whole domain Ω, we need to find a continuous extension of this velocity which gives the front velocity as output at the position of the front. In this work, we take this continuous extension to be simply the constant function $F(x, t^n) = v^n, x \in \Omega$. Note that this extension is very elegant in one dimension. However, if we want to solve the Stefan problem in more dimensions, this extension is not directly applicable.

6.4 Discretization of the advection problem

In this section, we will derive a numerical procedure which can be followed to solve the advection problem. In order to solve the advection equation, we need the continuous extension of the moving boundary velocity. This velocity describes how fast the level-set function $\phi(x,t)$ moves. We will show that the second order accurate Lax-Wendroff method applied to the one-dimensional signed distance function reduces to an explicit, first order scheme which only involves time step size. Using this scheme as a base, we will construct a second order accurate scheme by the means of Lagrange polynomials.

Suppose that we apply the Lax-Wendroff method to the advection equation $\phi(x,t)$. This explicit method first approximates, given the advection velocity v^n, the function at time $t = t^{n+1/2}$. Then, by using the central differences method, the temperature at time $t = t^{n+1}$ can be approximated. We can express this in the following equations:

$$\phi_{i-1/2}^{n+1/2} = \frac{1}{2}(\phi_i^n + \phi_{i-1}^n) + \frac{v^n \Delta t}{2\Delta x}(\phi_i^n - \phi_{i-1}^n) \tag{6.11}$$

$$\phi_{i+1/2}^{n+1/2} = \frac{1}{2}(\phi_i^n + \phi_{i+1}^n) + \frac{v^n \Delta t}{2\Delta x}(\phi_{i+1}^n - \phi_i^n) \tag{6.12}$$

$$\phi_i^{n+1} = \phi_i^n - \frac{v^n \Delta t}{\Delta x}(\phi_{i+1/2}^{n+1/2} - \phi_{i-1/2}^{n+1/2}) \tag{6.13}$$

This scheme can also be written as

$$\phi_i^{n+1} = \frac{\alpha}{2}(\alpha+1)\phi_{i-1}^n + (1-\alpha^2)\phi_i^n + \frac{\alpha}{2}(\alpha-1)\phi_{i+1}^n \tag{6.14}$$

where $\alpha = \frac{v^n \Delta t}{\Delta x}$ is the CFL-number for the advection problem. This method is stable for $\alpha \leq 1$. Now, we use some handy properties of the one-dimensional signed distance function to simplify this expression. First, note that the spatial derivative of $\phi(x,t)$ can be easily derived to be

$$\frac{\partial \phi}{\partial x}(x,t) = -1 \tag{6.15}$$

for all $x \in \Omega, t \geq 0$. This can simply be seen from the fact that the one-dimensional signed distance function is a linear function. But, in that case, we also know that the numerical forward, backward and centeal differences estimates for the derivative are exactly equal to the analytical derivative of $\phi(x,t)$, i.e.:

$$\frac{\phi_{i+1}^n - \phi_{i-1}^n}{2\Delta x} = \frac{\phi_{i+1}^n - \phi_i^n}{\Delta x} = \frac{\phi_i^n - \phi_{i-1}^n}{\Delta x} = -1 \tag{6.16}$$

We also know that the second spatial derivative of the signed distance function equals 0. For the central difference approximation of the second spatial derivative, it also holds that this numerical estimate exactly equals 0, since $\phi(x,t)$ is a linear function:

$$\frac{\phi_{i-1}^n - 2\phi_i^n + \phi_{i+1}^n}{\Delta x^2} = 0 \tag{6.17}$$

We can rewrite the Lax-Wendroff scheme:

$$\frac{\alpha^2}{2}(\phi_{i-1}^n - 2\phi_i^n + \phi_{i+1}^n) + \phi_i^n - \frac{\alpha}{2}(\phi_{i+1}^n - \phi_{i-1}^n) = \phi_i^{n+1} \tag{6.18}$$

We recognize in this form the central differences approximation of the first and second derivatives. Substituting the also-known exact values for this points, we obtain the following simplified scheme:

$$\phi_i^{n+1} = \phi_i^n + \Delta t v(t^n) \tag{6.19}$$

This scheme is first order accurate in time. Note that the error due to the spatial step size Δx has disappeared: the whole numerical error is a function of Δt. So, what we have shown is that the Lax-Wendroff method applied to a linear function like the signed distance function simply reduces

to a first order accurate scheme of which the numerical error only depends on the time step size. However, in this work we attempt to reach second order accuracy. Therefore, we need at least a second order accurate method to solve this equation. We do this by using a weighted velocity to update $\phi(x,t)$:

$$\phi_i^{n+1} = \phi_i^n + \frac{\Delta t}{2}(v(t^n) + v(t^{n+1})) \tag{6.20}$$

Note that this is an implicit scheme. The problem of this scheme is that we don't know the velocity at time $t = t^{n+1}$. Therefore, we need to make an estimate of this velocity. This can be done by constructing a second order Lagrange polynomial which estimates the velocity:

$$P_2(t) \approx v^{n-2}\frac{(t-t^{n-1})(t-t^n)}{(t^{n-2}-t^{n-1})(t^{n-2}-t^n)} + v^{n-1}\frac{(t-t^{n-2})(t-t^n)}{(t^{n-1}-t^{n-2})(t^{n-1}-t^n)} + v^n\frac{(t-t^{n-2})(t-t^{n-2})}{(t^n-t^{n-2})(t^n-t^{n-1})} \tag{6.21}$$

If we fill in $t = t^{n+1}$ in this equation, we find the following estimate of v^{n+1}:

$$v^{n+1} \approx v^{n-2} - 3v^{n-1} + 3v^n \tag{6.22}$$

Using this estimate of v^{n+1}, we find the following scheme for updating the advection equation for $n \geq 2$.

$$\phi_i^{n+1} = \phi_i^n + \frac{\Delta t}{2}(v^{n-2} - 3v^{n-1} + 4v^n) \tag{6.23}$$

For $t = t^0$ and $t = t^1$, this scheme is not applicable. For these time steps, we can simply use the first order accurate scheme $\phi_i^{n+1} = \phi_i^n + \Delta t v(t^n)$. Another option for $t = t^1$ is to estimate v^{n+1} with a first order Lagrange polynomial. In the Results, we will investigate if the derived method indeed leads to second order accurate results.

The described method leads to a procedure where re-initialization of the signed distance function is not necessary, since all points in the domain are linearly shifted with velocity v^n. Therefore, in this case we do not need a re-initialization procedure. However, note that this may be necessary when we want to extend this method to higher dimensions.

6.5 Discretization of the heat problem

The last procedure which needs to be performed to solve the Stefan problem, is updating the heat equation (with taking in consideration the Stefan condition at the moving boundary). In this section, we will derive the procedure to update this equation numerically.

The heat equation in the grid points in the interior of the domain is discretized using a Crank-Nicolson scheme, since this numerical method is second order accurate in space and time. For the gridpoints next to the moving boundary, we use a Lagrange polynomial to estimate the temperature in these gridpoints. We call these gridpoints x_j and x_{j+1}. The Crank Nicolson for equation for $j = 2, \ldots, j-1$ and $j+2, \ldots M-1$ is given by

$$\frac{\hat{T}_i^{n+1} - \hat{T}_i^n}{\Delta t} = \frac{k_i}{2}\left(\frac{\hat{T}_{i+1}^{n+1} - 2\hat{T}_i^{n+1} + \hat{T}_{i-1}^{n+1}}{\Delta x^2} + \frac{\hat{T}_{i+1}^n - 2\hat{T}_i^n + \hat{T}_{i-1}^n}{\Delta x^2}\right) \tag{6.24}$$

where k_i denotes the thermal diffusivity, which is equal to k_L in the liquid domain and k_S in the solid domain. Rewriting this equation and introducing $\alpha_i = \frac{k_i \Delta t}{\Delta x^2}$ leads to:

$$-\alpha_i\hat{T}_{i+1}^{n+1} + (2+2\alpha_i)\hat{T}_i^{n+1} - \alpha_i\hat{T}_{i-1}^{n+1} = \alpha_i\hat{T}_{i+1}^n + (2-2\alpha_i)\hat{T}_i^n + \alpha_i\hat{T}_{i-1}^n \tag{6.25}$$

At the boundary points x_1, x_M, we have to apply the Dirichlet boundary condition. In order to do so, we introduce the virtual points $x_0 = -\frac{1}{2}\Delta x$ and $x_M = l + \frac{1}{2}\Delta x$. We know that the value of the boundary $x = 0$ at time $t = t^n$ is equal to $g_L(t^n) := T(0, t^n)$. Therefore, we can estimate

$$\frac{\hat{T}_1^n + \hat{T}_0^n}{2} = g_L(t^n) + \mathcal{O}(\Delta x^2) \tag{6.26}$$

Using this, we can express $\hat{T}_0^n$ in terms of known variables:

$$\hat{T}_0^n = 2g_L(t^n) - \hat{T}_1^n + \mathcal{O}(\Delta x^2) \tag{6.27}$$

Filling this in in the Crank-Nicolson scheme, we obtain

$$\frac{\hat{T}_1^{n+1} - \hat{T}_1^n}{\Delta t} = \frac{k_L}{2}\left(\frac{\hat{T}_2^{n+1} - 3\hat{T}_1^{n+1} + 2g_L(t^{n+1})}{\Delta x^2} + \frac{\hat{T}_2^n - 3\hat{T}_1^n + 2g_L(t^n)}{\Delta x^2}\right). \tag{6.28}$$

We can write this expression to a form with all known terms at the right hand side and all unknown terms at the left hand side. Note that we know the values of $g_L(t^n)$ and $g_L t^{n+1})$ since this function is a known boundary condition.

$$-\alpha\hat{T}_2^{n+1} + (2+3\alpha)\hat{T}_1^{n+1} = \alpha\hat{T}_2^n + (2-3\alpha)\hat{T}_1^n + 2\alpha g_L(t^n) + 2\alpha g_L(t^{n+1}) \tag{6.29}$$

A similar estimate and derivation yields the following expression for the right boundary

$$-\alpha\hat{T}_{M-1}^{n+1} + (2+3\alpha)\hat{T}_M^{n+1} = \alpha\hat{T}_{M-1}^n + (2-3\alpha)\hat{T}_M^n + 2\alpha g_S(t^n) + 2\alpha g_S(t^{n+1}) \tag{6.30}$$

where $g_S(t^N) = T(l, t^n)$.

Now, we have to estimate the temperature in points close to the moving boundary. In order to find these values, we use Lagrange polynomials to estimate the second spatial derivative. Using quadratic Lagrange polynomials and the assumption that the melting temperature at the moving boundary is given by $T_m = 0$, we estimate the temperature around the moving boundary from the left side as

$$T(x, t^n) \approx T_{j-1}^n \frac{(x - x_j)(x - s^n)}{(x_{j-1} - x_j)(x_{j-1} - s^n)} + T_j^n \frac{(x - x_{j-1})(x - s^n)}{(x_j - x_{j-1})(x_j - s^n)} \tag{6.31}$$

Taking the double derivative of this equation and using the definitions of r_1 and r_2, we can write

$$T_{xx}(x, t^n) \approx \frac{2T_{j-1}^n}{(\Delta x)^2(1+r_1)} - \frac{2T_j^n}{(\Delta x)^2 r_1} \tag{6.32}$$

Using this estimate for the double derivative, we find the following discretization for the heat equation by applying the Backward Euler method:

$$\frac{\hat{T}_j^{n+1} - \hat{T}_j^n}{\Delta t} = k_L\left(\frac{2\hat{T}_{j-1}^{n+1}}{(\Delta x)^2(1+r_1)} - \frac{2\hat{T}_j^{n+1}}{(\Delta x)^2 r_1}\right) \tag{6.33}$$

which is equivalent to

$$\hat{T}_j^n = \left(1 + \frac{2k_L\Delta t}{r_1\Delta x^2}\right)\hat{T}_j^{n+1} - \frac{k_l\Delta t}{\Delta x^2}\frac{2}{1+r_1}\hat{T}_{j-1}^{n+1} \tag{6.34}$$

For the grid point next to the moving boundary at the right hand side, we find in a similar way

$$\hat{T}_{j+1}^n = \left(1 + \frac{2k_s\Delta t}{r_2\Delta x^2}\right)\hat{T}_{j+1}^{n+1} - \frac{k_s\Delta t}{\Delta x^2}\frac{2}{1+r_2}\hat{T}_{j+2}^{n+1} \tag{6.35}$$

Combining the estimate for the boundary conditions and the discretized heat equation, this leads to the following matrix equation:

$$A\hat{T}^{n+1} = R^n \tag{6.36}$$

with

$$A = \begin{pmatrix} 2+3\alpha_L & -\alpha_L & & & & & & \\ -\alpha_L & 2+2\alpha_L & -\alpha_L & & & & & \\ & & \ddots & & & & & \\ & & -\frac{2\alpha_L}{1+r_1} & 1\frac{2\alpha_L}{r_1} & & & & \\ & & & & 1+\frac{2\alpha_S}{r_2} & -\frac{\alpha_S}{1+r_2} & & \\ & & & & & \ddots & & \\ & & & & & -\alpha_S & 2+2\alpha_S & -\alpha_S \\ & & & & & & -\alpha_S & 2+3\alpha_S \end{pmatrix} \tag{6.37}$$

and $\hat{T}^{n+1} = (\hat{T}_1^{n+1}, \hat{T}_2^{n+1}, \ldots, \hat{T}_{M-1}^{n+1}, \hat{T}_M^{n+1})^T$. Finally, R_n is given by

$$R^n = \begin{pmatrix} \alpha_L \hat{T}_2^n + (2-3\alpha_L)\hat{T}_1^n + 2\alpha_L g_L(t^n) + 2\alpha_L g_L(t^{n+1}) \\ \alpha_L \hat{T}_3^n + (2-2\alpha_L)\hat{T}_2^n + \alpha_L \hat{T}_1^n \\ \vdots \\ \hat{T}_j^n \\ \hat{T}_{j+1}^n \\ \vdots \\ \alpha_S \hat{T}_M^n + (2-2\alpha_S)\hat{T}_{M-1}^n + \alpha_S \hat{T}_{M-2}^n \\ \alpha_S \hat{T}_{M-1}^n + (2-3\alpha_S)\hat{T}_M^n + 2\alpha_S g_S(t^n) + 2\alpha_S g_S(t^{n+1}) \end{pmatrix} \tag{6.38}$$

6.6 Discretizing the numerical error

Having defined a procedure to solve the Stefan problem numerically, we want to be able to find out how accurate this method is. We do this by comparing the numerical and the analytical solution for the moving boundary position for different numbers of time and spatial steps. In order to do so, we have to choose some time step $t = t^N$ at which we compare the numerical error. Suppose we fix the number of time steps at N, which means that $\Delta t = \frac{t^N - t^0}{N}$. Than the numerical L^2-error due to spatial step size at time step $t = t^N$ is given by

$$\textit{error due to spatial step size} = \frac{1}{\sqrt{N}} (\sum_{n=0}^{N} s^n)^{1/2} \tag{6.39}$$

where s^n denotes the numerical solution to the Stefan problem at time $t = t^n$. The formula for the numerical error due to time step size looks exactly similar. By storing the numerical error for different values of the numbers of space and step sizes, we can investigate how the numerical error changes. In this work, we strive for second order accuracy. This means that the numerical error will decrease in a quadratic way when we linearly decrease the number of time steps or the number of spatial steps. However, if we fix the number of time steps N and decrease the number of spatial steps M, we will see that the numerical error reaches a certain limit. This limit is the numerical error caused by the fixed number of time steps N. A similar problem occurs when we fix M and vary N. To avoid this problem, we can choose the time step size as a linear function of the spatial step size, $\Delta t = a\Delta x$, with $a \in \mathbb{R}_{>0}$. If the numerical solution is second order accurate then for as well space as time, we then have that the error is of order $\mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t^2) = \mathcal{O}(\Delta x^2) + \mathcal{O}(a^2 \Delta x^2) = \mathcal{O}(\Delta x^2)$. Therefore, the error will continuously decrease in this case with second order if we vary the number of spatial steps.

7 Numerical results and discussion

In this chapter, the results of the discretization of the Stefan problem as described in the previous chapter are presented. In the first section (7.1), we investigate the accuracy of the developed method for the heat problem without moving boundary. In the following section (7.2), we consider the accuracy of the developed method for the advection problem with given analytical velocity function. In the last section of this chapter, (7.3), we solve the Stefan problem with the described level-set method and we investigate the accuracy of this solution to find out if it reaches second order accuracy.

7.1 Heat problem without moving boundary

In this section, we consider the Crank-Nicolson scheme introduced in the previous chapter as solution for the heat equation with non-homogeneous Dirichlet boundary conditions. Note that we do not yet consider the heat problem with moving boundary (the Stefan problem) here - first we test our level-set method on a heat problem without phase transitions with fixed boundaries, where the boundary temperature is prescribed by the similarity solution to the Stefan problem. We choose as domain $\Omega = [0, 0.5]$, for the thermal diffusivities we take $k_L = k_S = 1$ and we also let $L = \rho = 1$. As initial temperature distribution, we choose $T_i^0 = T_{sim}(x_i, t = 0.001)$, with $T_{sim}(x, t)$ the similarity solution. As parameters for this solution, we choose $s_0 = 0.55, T_L = 0.53$ and $T_S = -0.1$. Note that in our domain, the whole domain Ω is in liquid phase for all times $t \geq 0$, since the moving boundary moves to the right in the chosen Stefan problem and the initial position of the moving boundary is right to our domain. For the sake of clarity, we give the Crank Nicolson scheme for the inner boundary points $i = 2, \ldots, i = M - 1$ again:

$$-\alpha_i \hat{T}_{i+1}^{n+1} + (2 + 2\alpha_i)\hat{T}_i^{n+1} - \alpha_i \hat{T}_{i-1}^{n+1} = \alpha_i \hat{T}_{i+1}^n + (2 - 2\alpha_i)\hat{T}_i^n + \alpha_i \hat{T}_{i-1}^n \tag{7.1}$$

At the boundary points x_1, x_m, we use the Dirichlet boundary conditions to find the following equation for the left boundary $x = x_0$:

$$-\alpha \hat{T}_2^{n+1} + (2 + 3\alpha)\hat{T}_1^{n+1} = \alpha \hat{T}_2^n + (2 - 3\alpha)\hat{T}_1^n + 2\alpha g_L(t^n) + 2\alpha g_L(t^{n+1}) \tag{7.2}$$

For the right boundary, we find

$$-\alpha \hat{T}_{M-1}^{n+1} + (2 + 3\alpha)\hat{T}_M^{n+1} = \alpha \hat{T}_{M-1}^n + (2 - 3\alpha)\hat{T}_M^n + 2\alpha g_S(t^n) + 2\alpha g_S(t^{n+1})$$

(7.3)

Since the moving boundary velocity is positive, the moving boundary goes to the right direction. Therefore, the phase of the domain Ω stays liquid for all times $t \geq t^0 = 0.001$. Since we want to start with a smooth function at our initial time step, we start our calculations at $t^0 = 0.001$ and we calculate the evolution of the temperature between $t = 0.001$ and $t = 0.1$ for different numbers of spatial and temporal step sizes. We compare the numerical error for several simulations at time $t = 0.1$. In this whole chapter, we will compare the numerical solutions at this time. Note that the numerical L^2-error of the Crank-Nicolson scheme at this time is given by

$$||e||_2 = \frac{1}{\sqrt{M}}\left(\sum_{i=1}^{M}(T_i^N - T_{sim}(x_i, t = t^N))^2\right)^{1/2} \tag{7.4}$$

where T_i^N is the numerical solution at time $t = t^N = 0.1$.

In Figure 18, we show the analytical solution of the described problem at times $t = 0.001, t = 0.01, t = 0.03, t = 0.06, t = 0.1$. We see that the temperature for all these times is positive, so we indeed do not have to deal with the moving boundary in this domain for these times.

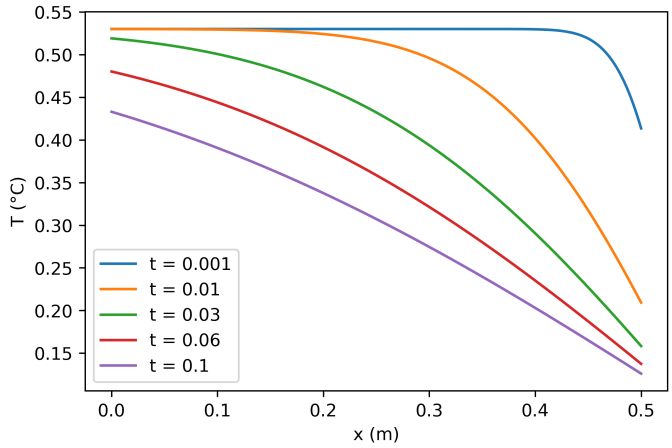


Figure 18: Analytical temperature profiles at times $t = 0.001, t = 0.01, t = 0.03, t = 0.06, t = 0.1$.

Using the given error formula, we can calculate the numerical error for different numbers of time steps N. In Figure 19, we see the numerical error of the temperature distribution at $t = 0.1$ as a function of the number of time steps with the number of spatial steps fixed at $M = 200$, $M = 400$, $M = 800$ and $M = 1600$.

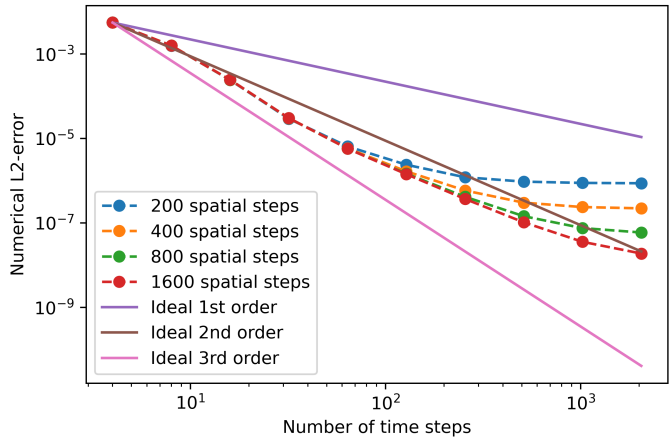


Figure 19: Log-log plot of the numerical error of the numerical solution to the heat problem as a function of the number of spatial steps. The numbers of time steps were fixed at $N = 200, N = 400, N = 800$ and $N = 1600$.

We see that this error is second order for lower numbers of time steps ($N < 100$): it follows the ideal second order line in this case. However, we see that the numerical error reaches a certain limit for each fixed number of spatial steps M. How larger the fixed number of spatial steps is, how smaller this limit is. This is probably caused by the fact that the (fixed) numerical error due to spatial step size is predominant for larger numbers of time steps: the numerical error due to the time step size becomes negligible with respect to the numerical error due to spatial step size.

We can also plot the numerical error as a function of the number of spatial steps. This plot is shown in Figure 20 for fixed numbers of time steps (namely $N = 5000, N = 10000$ and $N = 20000$).

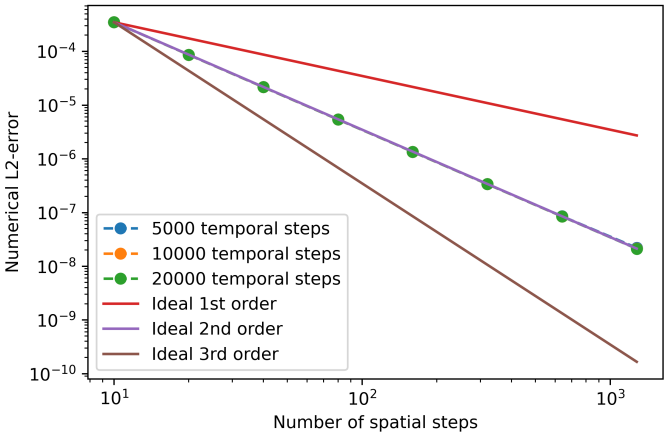


Figure 20: Log-log plot of the numerical error of the numerical solution to the heat problem as a function of the number of spatial steps. The numbers of spatial steps were fixed at $M = 250, M = 500, M = 1000$ and $M = 2000$.

In Figure 20, we see that the numerical error due to the spatial discretization follows the ideal second order for all investigated numbers of spatial steps. Therefore, we can conclude that the numerical error due to time step size for $N \geq 5000$) appears to be negligible compared to the spatial numerical error for $N \leq 1600$.

The last plot we make in this section is a plot where the time step size is given as a linear function of Δx, i.e.: $\Delta t = a\Delta x$, with $a \in \mathbb{R}_{\geq 0}$. If the given scheme is indeed of second order accuracy, which is indicated by the previous plots, we should also see second order convergence in this plot, since we have then that the following holds for the numerical error

$$\mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t^2) = \mathcal{O}(\Delta x^2) + \mathcal{O}(a^2 \Delta x^2) = \mathcal{O}(\Delta x^2) \tag{7.5}$$

In Figure 21, we see the numerical error as a function of the spatial step size, where the time step size is also chosen as a linear function of the spatial step size ($\Delta t = a\Delta x$). We see that the numerical error is of second order for both values of a. Since we chose the time step as a linear function of the spatial step size, it seems to be very plausible that this method is second order accurate, as we expected since the Crank-Nicolson is second order accurate in time and space.

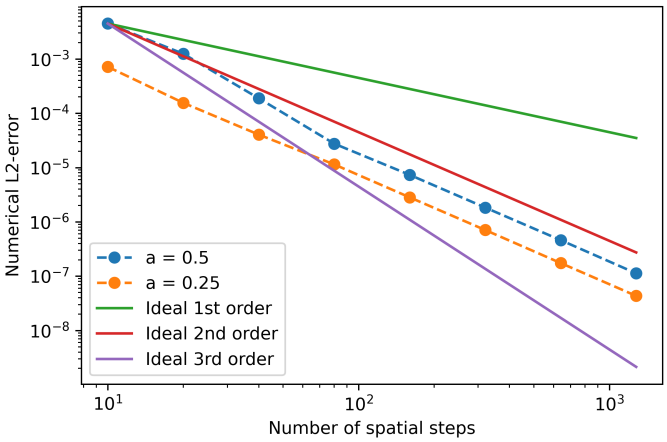


Figure 21: Log-log plot of the numerical error of the numerical solution to the heat problem as a function of the number of spatial steps. The time step was chosen as $\Delta t = a\Delta x$, with $a \in \{1/4, 1/2\}$.

7.2 Advection problem with analytical velocity

In this section, we will investigate the advection problem and the accuracy of the developed method to solve this problem. We consider it separately from the Stefan problem. In order to do so, we consider the advection problem with given analytical velocity of the moving boundary $v(t)$. Note that this velocity for the Stefan problem is analytically given by the temporal derivative of the position of the moving boundary $s(t)$:

$$v(t) = \frac{ds}{dt} = \frac{d}{dt}(s(0) + 2\lambda\sqrt{t}) = \frac{\lambda}{\sqrt{t}} \tag{7.6}$$

Recall that the numerical scheme used to solve the advection problem in this work is given by

$$\phi_i^{n+1} = \phi_i^n + \frac{\Delta t}{2}(v^{n-2} - 3v^{n-1} + 4v^n) \tag{7.7}$$

where v^n denotes the velocity at time $t = t^n$. In the numerical Stefan problem, we know that the velocities v^n, v^{n-1} and v^{n-2} are numerical estimates. However, in this section we use the analytical boundary velocity to compute these values (this is done in order to compare the advection problem separate from the heat problem). Note that the used scheme is not dependent of the spatial step size Δx, so we don't expect that the spatial step size influences the numerical error. To check this hypothesis, we show the numerical error as a function of the number of spatial steps in Figure 22.

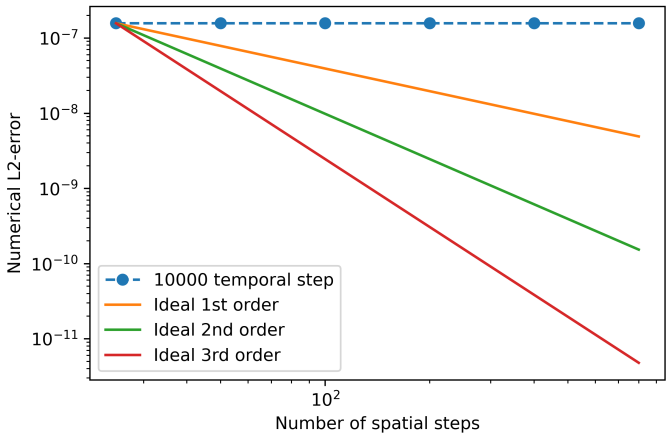


Figure 22: Log-log plot of the numerical error of the numerical solution to the heat problem as a function of the number of spatial steps. The number of time steps was fixed at $N = 10000$.

We indeed see in Figure 22 that the accuracy of our numerical solution to the advection problem is not dependent on the spatial step size: this error stays constant for different numbers of spatial steps. Now, we want to check the temporal accuracy of this numerical scheme. In Figure 23, we show the numerical error as a function of the time step size for fixed spatial step size. Recall that the numerical scheme in 7.7 is, as derived in the previous chapter, theoretically second order accurate in time. Note that it does not make sense to plot for more than one number of spatial steps M, since we just saw that the numerical error does not depend on the spatial step size. Therefore, it suffices to fix the number of spatial steps at, let's say, $M = 100$. This leads to the results presented in Figure 23.

We see that this error is indeed second order accurate in time. However, note that in this section we used the exact value of the boundary velocity $v(t)$ to fill in the scheme in 7.7. The big question after this section is if the numerical accuracy of this method will also be of second order when we use an estimation of the boundary velocity instead of the analytical velocity.

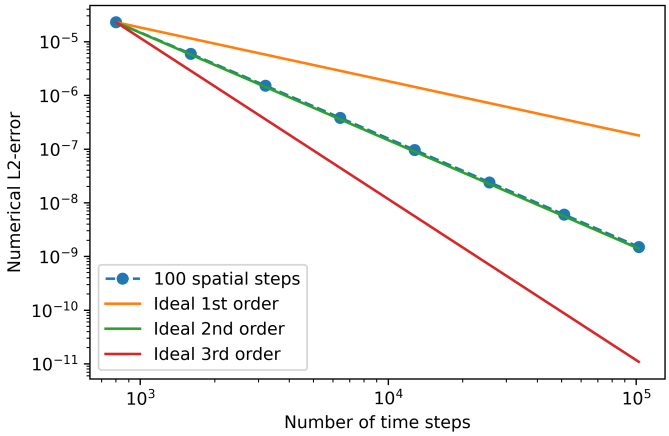
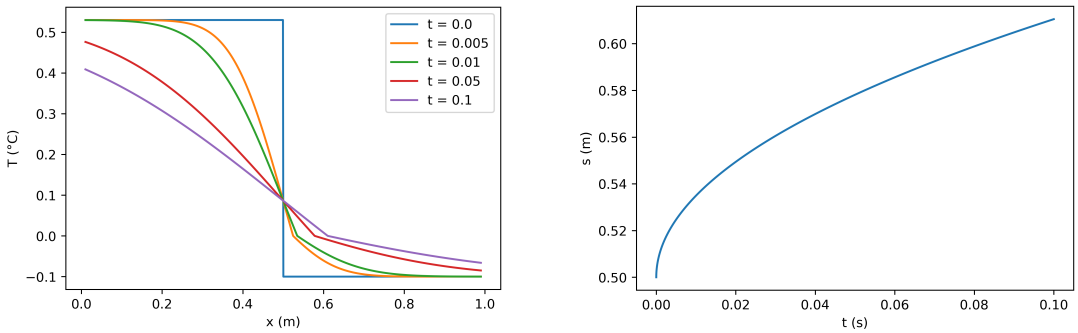


Figure 23: Log-log plot of the numerical error of the numerical solution to the heat problem as a function of the number of time steps. The number of spatial steps was fixed at $M = 100$.

7.3 Stefan problem

In this section, we present and discuss the results of the presented numerical method for solving the Stefan problem. The method described in Chapter 6 will be used to discretize the problem. This method is shortly summarized by the following: a second-order Lagrange polynomial is used to estimate the front velocity. After this, the signed distance function is updated with this velocity by solving the advection problem. Using the updated signed distance function, we update the temperature distribution in our domain with a Crank-Nicolson scheme for inner boundary points. For points close to the boundary, we use Lagrange polynomials to find the updated values.

We solve the Stefan problem in this work on domain $\Omega = [0, 1]$. As parameters for the problem, we choose initial boundary position $s(0) = 0.5$. For the sake of simplicity, we make some assumptions about the material's properties: we assume that they are constant in time and space. For the materials density, we choose $\rho = 1$, for the thermal diffusivities we choose $k_L = k_S = 1$ and we also let $L = 1$. As initial temperature distribution, we choose the similarity solution at $t = 0.001$, with initial parameters $T_L = 0.53$ and $T_S = -0.1$. We compare the numerical solution with the analytical (similarity) solution at $t = 0.1$ using the error formula given in Equation 7.4. The analytical temperature profile at several times including $t = 0.1$ is shown in Figure 24, as well as the similarity solution for the position of the moving boundary.



(a) Temperature profiles of the analytical solution at times $t = 0, t = 0.005, t = 0.01, t = 0.05$ and $t = 0.1$.

(b) Numerical error at time $t = 0.1$ for the FTCS-scheme for $\Delta t = 2.5 \cdot 10^{-5}$ and a varying spatial step size.

Figure 24: Similarity solution. In the left figure, temperature profiles at several times are shown. In the right figure, we position of the moving boundary is shown as a function of time. The parameters used to create these figures are $\Omega = [0, 1]$, $T_L = 0.53, T_S = -0.1, s_0 = 0.5, \rho = 1, L = 1, k_L = k_S = 1$.

First of all, we show the numerical error of this method as a function of the number of spatial steps for fixed numbers of time steps. This relation is shown in Figure 25.

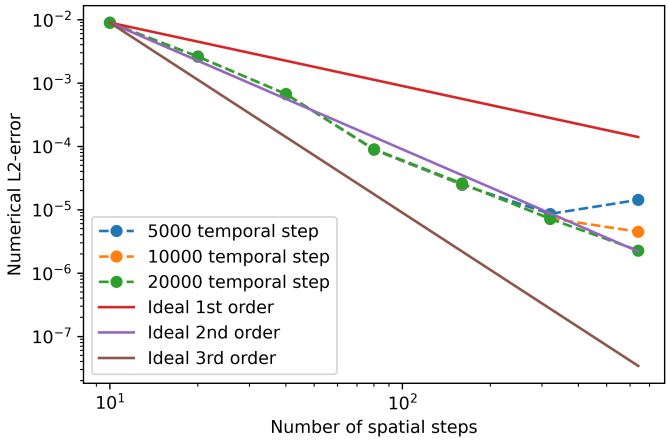


Figure 25: Log-log plot of the numerical error of the numerical solution to the heat problem as a function of the number of spatial steps. The number of time steps was fixed at $N = 5000, N = 10000, N = 20000$.

We see in this figure that the numerical error is approximately decreasing with second order for numbers of spatial steps between $M = 10$ and $M = 320$. For larger numbers of spatial steps, we see that the numerical error decreases further with second order for $N = 20000$ time steps, but that it stabilizes for $N = 10000$ time steps and for $N = 5000$ we even see that the numerical error increases. Later on, we will discuss possible causes for this behaviour, which only occurs when the time step size is relatively big compared to the spatial step size. Now, we take a look at the numerical error as a function of the number of time steps. This is shown in Figure 26.

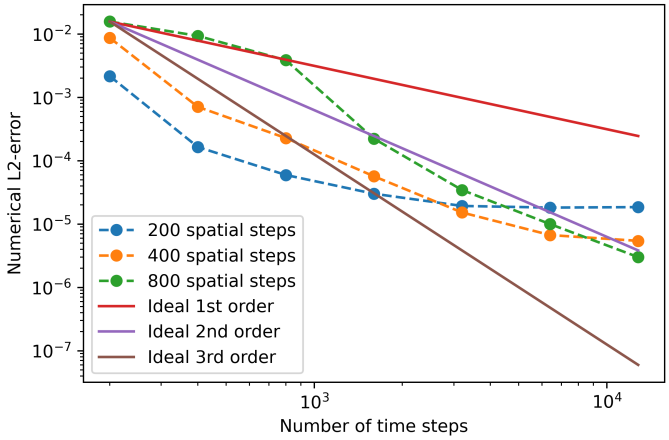
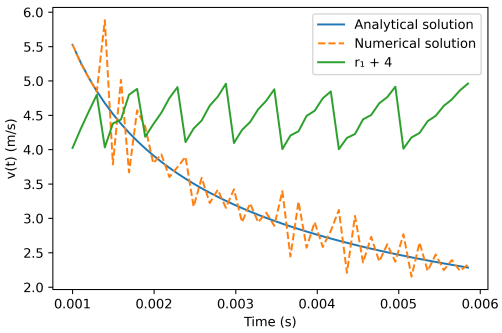
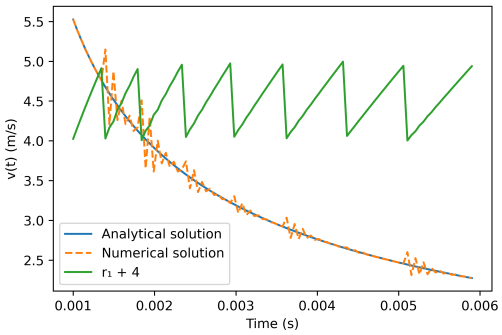


Figure 26: Log-log plot of the numerical error of the numerical solution to the Stefan problem as a function of the number of time steps. The numbers of spatial steps were fixed at $M = 200, M = 400$ and $M = 800$.

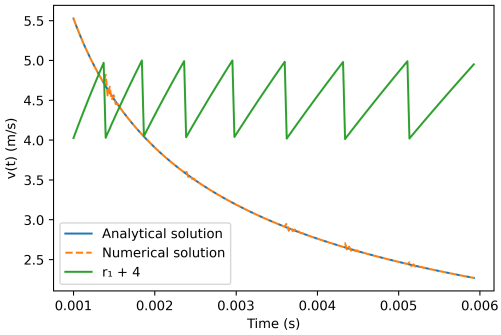
In Figure 26, we see that numerical error due to time step size is showing some non-trivial behaviour: for example when we look at the error for $M = 800$ spatial steps, we see that this error is of first order until we reach $N = 800$, then the error starts decreasing (approximately) following the ideal second order. We also observe that the numerical errors for $M = 200, M = 400$ seem to converge to a constant, fixed error limit. Possibly, this could be the limit of the fixed numerical error due to spatial step size. Something which could eventually explain the behaviour shown in Figure 26 is that the numerical model shows oscillatory behaviour when the time step size is relatively small compared to the spatial step size (which corresponds to the left half of Figure 26). In Figure 27, we show the numerically computed velocity versus the velocity computed using the similarity solution (namely $v(t) = \frac{ds}{dt} = \frac{\lambda}{\sqrt{t}}$) between times $t = 0.001$ and $t = 0.006$. Furthermore, the distance between the grid point x_j and the numerical position of the moving boundary s^n is shown by means of the (shifted) coefficient r_1.



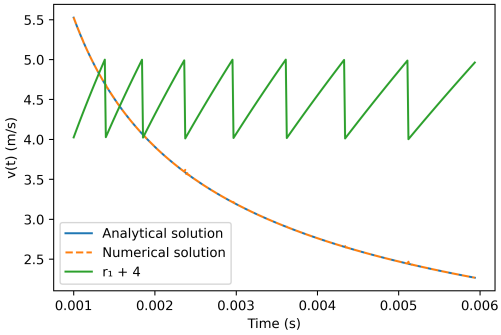
(a) $M = 500, N = 1000$



(b) $M = 500, N = 2000$



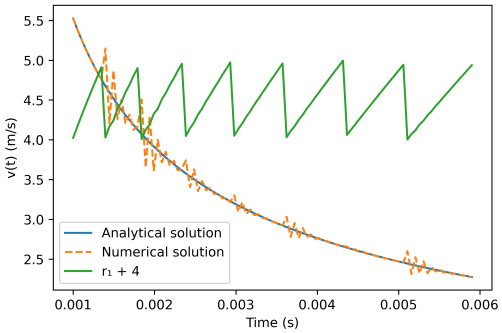
(c) $M = 500, N = 4000$



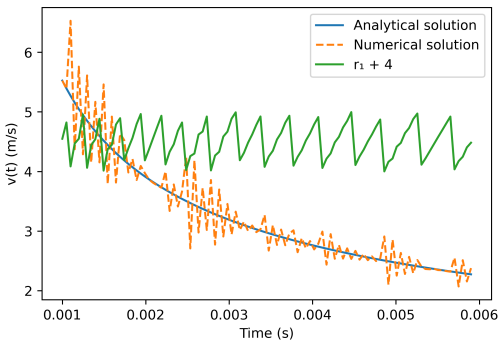
(d) $M = 500, N = 8000$

Figure 27: Numerical and analytical velocity as a function of time. M is equal for all figures: $M = 500$. The blue line represents the analytical solution of the boundary velocity $v(t)$, the dashed orange line represents the numerical solution. The (4 upwards shifted) coefficient r_1 is shown in green.

In Figure 27, we see that for $M = 500, N = 1000$ the estimate of the boundary velocity is oscillating a lot. For larger values of the number of time steps N, the oscillation decreases and we see that for $N = 8000$ the oscillation is no longer visible. Note that the largest oscillations always occur when the coefficient r_1 is close to a "jump" from 1 to 0, in other words: when the moving boundary is close to the grid point x_j or x_{j+1}. For $r_1 \geq 1/3$ and $r_2 \geq 1/3$, we use different Lagrange polynomials to estimate the boundary velocity than for the cases $r_1 > 1/3$ and $r_2 > 1/3$ (we do not include the points x_j and x_{j+1} in the first case). This could possibly lead to less accurate estimates for the velocity. The could be one of the causes for the oscillations shown in Figure 27.



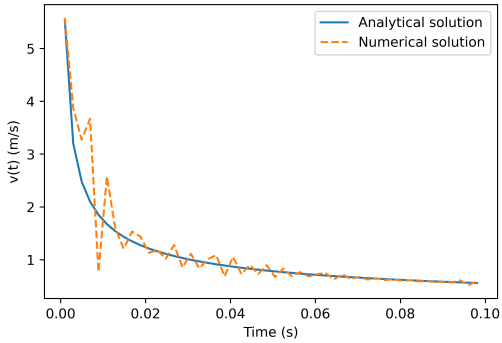
(a) $M = 500, N = 2000$



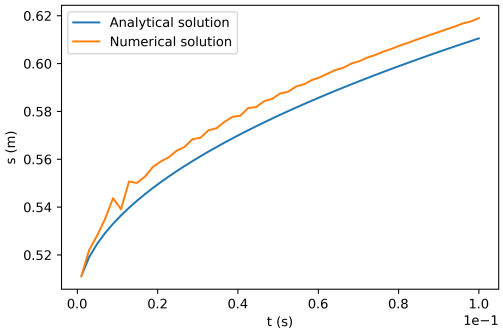
(b) $M = 1000, N = 2000$

Figure 28: Numerical and analytical velocity as a function of time. M is equal for all figures: $M = 500$. The blue line represents the analytical solution of the boundary velocity $v(t)$, the dashed orange line represents the numerical solution. The (4 upwards shifted) coefficient r_1 is shown in green.

In Figure 28, we plot the boundary velocity as a function of time for two values of spatial steps M ($M = 500, M = 1000$) and constant number of time steps N. What we see is that the number of oscillations is much higher for the case where $M = 1000$. What we also observe is that the number of 'jumps' of r_1 goes from 6 to 14. This is expected, since the number of grid points is doubled, which means that the moving boundary passes approximately twice as many grid points x_j. However, this doubling of grid points apparently also leads to more oscillations in the estimation of the velocity. If we want to compute with higher spatial accuracy therefore, we do need to take into consideration that the number of time steps also must be increases; otherwise our solution will become less accurate instead of more accurate. That oscillations in the boundary velocity lead to larger inaccuracies in the position of the moving boundary can easily be seen in Figure 29.



(a) Analytical and numerical velocity of the moving boundary for $M = 100, N = 50$.



(b) Analytical and numerical position of the moving boundary for $M = 100, N = 50$

Figure 29: Comparison of numerical and analytical solution to the Stefan problem for $M = 100, N = 50$. In the left figure, the velocity of the moving boundary is shown. In the right figure, the position of the moving boundary is plotted. Observe that at the times where larges oscillations in velocity occur (for example when $t = 0.01$), also oscillations in the moving boundary position are clearly visible.

So, reconsidering the behaviour of the numerical solutions to the level-set method when the time step size is small compared to the spatial step size, we came to the following observations.

1. The numerically computed boundary velocity, which makes use of Lagrange polynomials, shows large oscillations for high numbers of spatial steps M combined with low numbers of time steps N. These oscillations mainly occur when the moving boundary is in the proximity of grid points x_j or x_{j+1}.

2. The amount of oscillations increases when the number of spatial steps is increased. This is explainable by the fact that for higher numbers of spatial steps, more grid points x_j will be passed by the moving boundary. Since oscillations occur mostly when the boundary passes a grid point, more grid points will lead to more oscillations.

3. For a fixed number of spatial steps, the oscillations will decrease when the number of time steps is increased. The procedure to find the boundary velocity is not dependent of the time step size Δt. However, the time step is important for the update of the temperature $T(x, t)$ and the signed distance function $\phi(x, t)$, and both of these functions are used in the velocity estimate.

So, an important thing to consider when using the developed level-set method is which ratio between time step and spatial step should be chosen in order to avoid too large oscillations (since these will influence our numerical accuracy a lot). It is useful to figure out what the ratio between the time step size and spatial step size is where oscillations are negligible. In the ideal case, this ratio is linear (i.e. $\Delta t = a\Delta x$, $a \in \mathbb{R}_{\geq 0}$), because this would mean that, if our level-set method is second order accurate, we can in our simulation simply choose the time step size as a fraction of the spatial step size, and this would lead to second order accurate results.

Having considered temporal and spatial accuracy of our level-set method, we now want to consider the case where we choose the temporal time step as a function of the spatial step size: $\Delta t = a\Delta x$, with $a \in \mathbb{R}_{\leq 0}$. If our solution to the Stefan problem is second order accurate in as well space as time, we would expect that the numerical error as the function of the spatial step size would decrease with second order if we choose the time step as a linear function of the spatial step, since we have that

$$\mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t^2) = \mathcal{O}(\Delta x^2) + \mathcal{O}(a^2 \Delta x^2) = \mathcal{O}(\Delta x^2). \tag{7.8}$$

In Figure 30 we show the numerical error of our level-set method for different values of a ($a = 1/50, a = 1/100, a = 1/200$ and $a = 1/400$). The first thing we note is that the developed level-set method seems to follow the ideal second order for lower numbers of spatial steps. However, each of the shown lines starts to converge to a certain constant numerical error (for $a = 1/100$, this happens for example approximately after $M = 160$, and for $a = 1/200$ it starts to converge after $M = 640$).

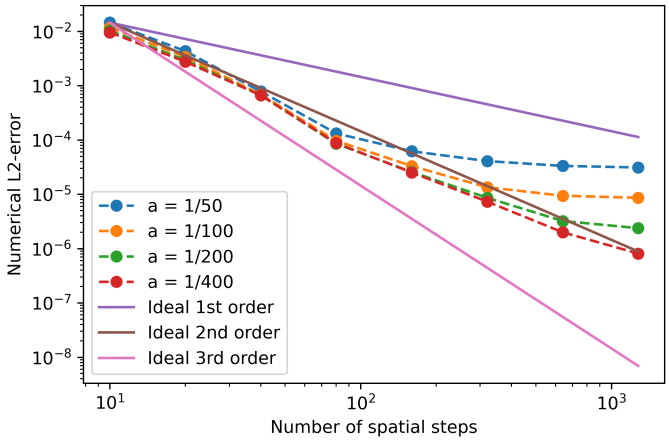
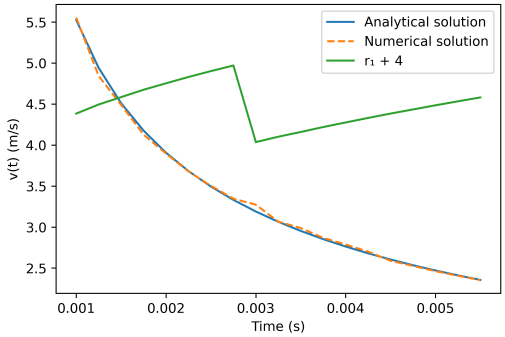


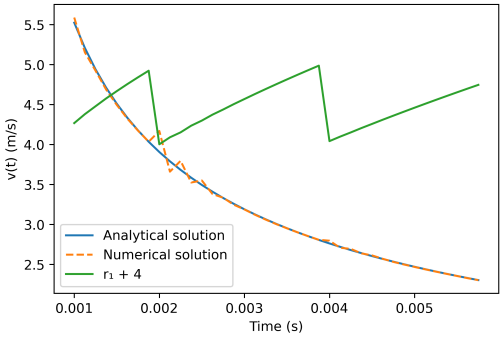
Figure 30: Log-log plot of the numerical error of the numerical solution to the Stefan problem as a function of the number of spatial steps. The time step was chosen as $\Delta t = a\Delta x$, with $a \in \{1/50, 1/100, 1/200, 1/400\}$.

Figure 30 shows interesting behaviour of our level-set model. If the numerical error converges to a constant, this means that our model does not become more accurate after a certain number of spatial steps: it does not further converge then, it is not even first order accurate - if the temporal or spatial error component was first order accurate, we would expect that the numerical error would follow the ideal first order line. One of the things which could maybe explain this constant numerical error is the oscillations in the velocity we discussed before. As we saw before, the error due to oscillations in the boundary velocity increases as the number of spatial steps increases. If this increasing oscillation error cancels out the decreasing error (caused by decreasing spatial step size) after a certain number of spatial steps, this could explain the results in Figure 30. To investigate this hypothesis, we show the numerical and analytical boundary velocity for $a = 1/50$ for $M = 80, M = 160, M = 320$ and $M = 640$ in Figure 31.

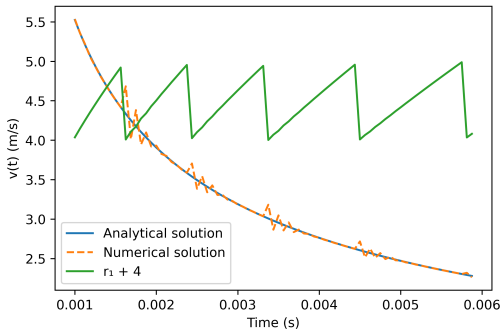
What we see in Figure 31 is that the influence of oscillations increases when the number of spatial steps M increases: in Figure 31a, we only see some small deviations from the analytical solution, while we see in Figure 31d many oscillations around the analytical solution. Since the spatial and temporal step size decrease, we expect that the Crank-Nicolson scheme and the scheme used to solve the advection equation will give more accurate solutions. However, as shown before, the numerical error becomes approximately constant after a certain number of spatial steps M. The increase in oscillatory behaviour of the boundary velocity estimate (which leads to increased numerical error) for larger numbers of spatial steps could cancel out the decrease in numerical error because of more accurate Crank-Nicolson and advection estimates.



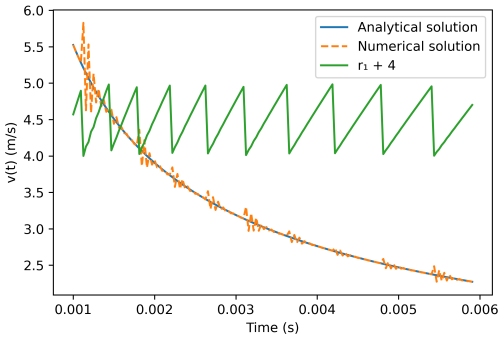
(a) $a = 1/50, M = 80$



(b) $a = 1/50, M = 160$



(c) $a = 1/50, M = 320$



(d) $a = 1/50, M = 640$

Figure 31: Numerical and analytical velocity as a function of time. a is equal for all figures: $a = 1/50$ while the number of spatial steps M is varying. The blue line represents the analytical solution of the boundary velocity $v(t)$, the dashed orange line represents the numerical solution. The (4 upwards shifted) coefficient r_1 is shown in green.

So, summarizing the results where Δt was a linear function of the spatial step, we can say that the influence of oscillations in the boundary velocity seems to play an important role here. The linear relation between time step size and spatial step size ($\Delta t = a\Delta x$) is apparently not sufficient to avoid these oscillations, since we still see here that the number of oscillations increases when the number of spatial steps increases. However, if we choose the value of a small enough (for example $a = 1/400$), the role of oscillations in the boundary velocity estimate is negligible for the whole investigated range of spatial steps ($10 \leq M \leq 1280$) and we see that the developed level-set method is approximately second order accurate for such small values of a.

Further research however is needed to establish the relation between the oscillations in the boundary velocity, the time step size Δt and the spatial step size Δx. In the first place it would be interesting to get to know more about conditions under which this oscillatory behaviour happens. Clearly, it is not sufficient to choose Δt as a linear small fraction of Δx, since oscillations will still be found then for large numbers of spatial steps. Further research could make clear if there exists a relation of the form $\Delta t = a\Delta x^2$ or some other polynomial relation between Δt and Δx which avoids large oscillations like we saw in this work.

Furthermore, future studies could look into alternative ways of estimating the moving boundary velocity. It seems to be the case that the estimation of this velocity is one of the biggest causes of the fact that our developed model is not completely of second order for certain numbers of spatial steps. Therefore, trying out other estimation procedures which involve extrapolation, like cubic splines, higher order Lagrange polynomials or Taylor polynomials, could be worth investigating.

Lastly, future research could focus on the question how to extend this method to more dimensions. The level-set method is because of its implicit nature relatively easy to extend to 2 or 3 dimensions. However, for higher dimensions, re-initialization of the signed distance function and the continuous extension of the velocity (which in this work was just chosen as a constant function) will need to be further considered before they can be extended to higher dimensions.

8 Conclusion

The purpose of this work was to develop a second order accurate level-set method to solve the two-phase Stefan problem with non-homogeneous Dirichlet boundary conditions. To do so, first the Stefan problem was described and analyzed. After this, a general introduction was given into numerical methods, the finite volume method and the level-set method. Using this knowledge, a level-set approach to the Stefan problem was described. In this approach, the Crank-Nicolson method was used to solve the heat equation. Lagrange polynomials were used to solve the heat equation around the moving boundary and to find the velocity of this boundary. Subsequently, the developed method was tested on its numerical accuracy in time and space.

Before testing the accuracy of the developed level-set method on the Stefan problem, the method was tested in two different parts. First, the accuracy of the temperature update procedure was investigated. It turned out that the developed level-set method was second order accurate in space and time when applied on a heat problem without moving boundary. This was expected, since the used level-set approach is basically a (second-order accurate) Crank-Nicolson scheme when applied to heat problem without moving boundary. Secondly, the procedure used to update the level-set function (the signed distance function) was considered. It appeared that the scheme used to solve the advection equation was second order accurate in time and independent of spatial step size when applied to a signed distance function with given velocity function. After this, the developed level-set method as a whole was applied to the Stefan problem. It was shown that the numerical estimate of the moving boundary velocity shows more oscillations and therefore becomes less accurate for high numbers of spatial steps. Increasing the number of time steps decreases the oscillations. All together, the developed level-set method showed second order accurate behaviour if the time step size was small compared to the spatial step size.

This study numerically solved the Stefan problem using a level-set approach. Where the goal was to reach second order accuracy, this goal is not fully achieved. To reach higher accuracy, this study recommends three things. In the first place, it seems to be promising to better investigate the estimation procedure of the moving boundary velocity. In this study, second order Lagrange polynomials were used to estimate this quantity. However, this method possibly caused large oscillations in the numerical solution when the moving boundary was close to a grid point. Therefore, alternative extrapolation procedures like cubic splines are worth considering. Secondly, this study recommends further research to the oscillatory behaviour of the estimation procedure of the moving boundary velocity, since this behaviour seems to be closely related to the non-converging behaviour of the numerical error for large numbers of spatial steps. What the relation is between time step size, spatial step size and oscillations in the moving boundary velocity estimate is a subject that deserves further study. Lastly, for the extension of the developed level-set method to higher dimensions, more research is needed. Especially the re-initialization of the signed distance function and the extension of the front velocity are things which are not easily extendable to more dimensions, but which will first need to be analyzed in further studies.

Acknowledgements

First and foremost, I would like to thank my first supervisor Prof. Dr. Ir. Kees Vuik and my second supervisor Prof. Dr. Ir. Danny Lathouwers for their feedback on my work. Each time I faced problems with the developed method or the sometimes inexplicable results, they asked me insightful questions which really helped me to find the errors in the developed method and to interpret the results. They helped me developing the project into a research direction I enjoyed and it is because of them that I am planning to continue my studies in the direction of numerical analysis. Furthermore, I would like to thank my housemates for lending an ear when I got stuck on some difficult piece of theory or when the Python model showed other results than I expected. I think my housemates have seen more temperature profiles and moving boundary plots in the past few months then they will ever see again. Lastly, I would like to thanks my parents, brothers and sisters for hearing my bachelor project stories every weekend I was at home. I could not have done this project without you.

References

Betounes, D. (1998). The 1-D Heat Equation. *Partial Differential Equations for Computational Science*, 35–51. doi: 10.1007/978-1-4612-2198-2_3

Chen, S., Merriman, B., Osher, S., & Smereka, P. (1997). A simple level set method for solving Stefan problems. *Journal of Computational Physics*, *135*(1), 8–29.

Eymard, R., Gallouët, T., & Herbin, R. (2000a). Finite volume methods. *Handbook of numerical analysis*, *7*, 713–1018.

Eymard, R., Gallouët, T., & Herbin, R. (2000b). Finite volume methods. *Handbook of numerical analysis*, *7*, 713–1018.

Gupta, S. (2018). *Chapter 1 - the Stefan problem and its classical formulation* (Second ed.; S. Gupta, Ed.). Amsterdam: Elsevier. doi: https://doi.org/10.1016/B978-0-444-63581-5.00001-4

Hill, J. M. (1987). *One-dimensional Stefan problems: an introduction* (Vol. 31). Longman Sc & Tech.

Holmes, M. (2010). Introduction to numerical methods in differential equations. , 127–154.

Javierre, E., Vuik, C., Vermolen, F., & van der Zwaag, S. (2006). A comparison of numerical models for one-dimensional Stefan problems. *Journal of Computational and Applied Mathematics*, *192*(2), 445-459. doi: https://doi.org/10.1016/j.cam.2005.04.062

Jonsson, T. (2013). On the one-dimensional Stefan problem: with some numerical analysis..

Kumar, M., & Mishra, G. (2011). An Introduction to Numerical Methods for the Solutions of Partial Differential Equations. , *2011*(November), 1327–1338. doi: 10.4236/am.2011.211186

Roh, W., & Kikuchi, N. (2002). Analysis of Stefan problem with level set method. In *8th aiaa/asme joint thermophysics and heat transfer conference* (p. 2874).

Voller, V., & Cross, M. (1981). Accurate solutions of moving boundary problems using the enthalpy method. *International Journal of Heat and Mass Transfer*, *24*(3), 545-556. doi: https://doi.org/10.1016/0017-9310(81)90062-4

Vuik, C., Vermolen, F., Gijzen, M., & Vuik, M. (2015). *Numerical methods for ordinary differential equations*. DAP, Delft Academic Press.

Appendix A: Python code

For numerically solving the Stefan problem and all related numerical problems, lots of Python codes have been generated. The interested reader can access this code via https://github.com/corne00/StefanProblem.

One of the most important pieces of code, which is used to obtain the relation between the numerical error and the number of spatial steps M, with $\Delta t = a\Delta x$ is shown below.

```
# Stefan problem modelled with the level-set method

### Import necessary modules
import numpy as np
import time as time
import matplotlib.pyplot as plt
from scipy import interpolate
from scipy.special import erf, erfc
from scipy.optimize import fsolve, curve_fit
from scipy.interpolate import lagrange
from scipy import linalg as linalg

### Set simulation parameters
xmin, xmax = 0.0, 1.0 # domain
tmin, tmax = 0.001, 0.1  # start and end time.
M_array = np.array([10,20,40,80, 160, 320, 640]) # numbers of spatial steps
a_array = np.array([1/25, 1/50])  # values of a (to compute dt = a*dx)
dx_array = (xmax-xmin)/M_array # values of dx
errors = np.zeros((len(M_array), len(a_array)), dtype = np.double) #  errors

### Problem parameters
s0 = 0.5 # initial position of the moving boundary
rho = 1.0 # materials density
L = 1.0 # latent heat
ks, kl = 1.0, 1.0 # heat diffusivities of liquid and solid
Tl, Ts, Tm = 0.53, -0.1, 0.0 #initial temperatures

### Define functions
def Transcendental(lam):
    """This function is necesessary to find the lambda which is needed to find the
        analytical solutions to the problem"""
    left = lam
    right = np.sqrt(ks)/(np.sqrt(np.pi)*L)*Ts/erfc(lam/np.sqrt(ks))*np.exp(-lam**2/
        ks)+np.sqrt(kl)/(np.sqrt(np.pi)*L)*Tl/(2-erfc(lam/np.sqrt(kl)))*np.exp(-lam
        **2/kl)
    return right-left
def T_analytical(lam,x,t):
    """This function returns the temperature at a given array of positions at a
        given time."""
    s = s0 + 2*lam*np.sqrt(t)
    T_ana = np.ones_like(x)*Tm
    T_ana = np.where(x<s,-Tl*erfc(lam/np.sqrt(kl))/(2-erfc(lam/np.sqrt(kl)))+Tl*
        erfc((x-s0)/(2*np.sqrt(kl*t)))/(2-erfc(lam/np.sqrt(kl))),T_ana)
    T_ana = np.where(x>s, Ts - Ts*erfc((x-s0)/(2*np.sqrt(ks*t)))/(erfc(lam/np.sqrt(
        ks))), T_ana)
    return T_ana
def s_analytical(lam, t):
    """This function returns the position of the moving boundary at given time"""
    s = s0 + 2*lam*np.sqrt(t)
    return s
def find_vel(Ti, xi, phi, dx):
    """This function estimates the velocity of the moving boundary by estimating
        the temperature gradients at both sides of the boundary and by using the
        Stefan condition"""
    Tgradsol = 0 # initial estimate for the gradient at the solid side
    Tgradliq = 0 # inital estimate for the gradient at the liquid side
    j = np.min(np.where(phi<0))-1 # find the index of the grid point in front of
        the moving boundary
    r1 = -(phi[j]/(phi[j+1]-phi[j])) # find the distance r1 between x_j and the
        moving interface (s - x_j = r1*dx)
    r2 = (phi[j+1]/(phi[j+1]-phi[j])) # find the distance r1 between x_{j+1} and
        the moving interface (x_{j+1} - s = r2*dx)
```

```
52      xs = xi[j] + r1*dx # find position of the moving boundary
53      # Find a second-order Lagrange polynomial and take its derivative to find the
            gradient jump at the moving boundary.
54      pc=1/3
55      if r1 <= pc:
56          Tgradliq = np.polyder(lagrange([xi[j-2], xi[j-1], xs], [Ti[j-2], Ti[j-1],
                Tm]))(xs)
57      elif r1 > pc:
58          Tgradliq = np.polyder(lagrange([ xi[j-1], xi[j], xs], [Ti[j-1], Ti[j], Tm])
                )(xs)
59      if r2 > pc:
60          Tgradsol = np.polyder(lagrange([xs, xi[j+1], xi[j+2]], [Tm, Ti[j+1], Ti[j
                +2]]))(xs)
61      elif r2 <=pc:
62          Tgradsol = np.polyder(lagrange([xs, xi[j+2], xi[j+3]], [Tm, Ti[j+2], Ti[j
                +3]]))(xs)
63      return (1/L*(ks*Tgradsol - kl*Tgradliq))
64  def solve_advection(vels, phi, M, dx, dt):
65      """This function solves the advection equation for the given array of constant
            velocities. It returns the updated advection function."""
66      if len(vels)==1:
67          phi_new = phi + dt*vels[-1]
68      elif len(vels)==2:
69          phi_new = phi + dt/2*(3*vels[-1]-vels[-2])
70      else:
71          phi_new = phi + dt/2*(vels[-3]-3*vels[-2]+ 4*vels[-1])
72      return phi_new
73  def update_temp(phi, Ti, t, lam, M, dx, dt):
74      """This function updates the temperature profile using the values of the signed
             distance function and the previous temperature at each grid point. This
            function used the FVM-CN scheme for points away from the boundary. Close to
             the boundary, Lagrange interpolation polynomials are used to obtain an
            estimate for the value close to the boundaries using a BE scheme."""
75      j = np.min(np.where(phi<0))-1 # find the index of gridpoint in front of the
            moving boundary
76      r1 = -(phi[j]/(phi[j+1]-phi[j])) # find the distance r1 between x_j and the
            moving interface (s - x_j = r1*dx)
77      r2 = (phi[j+1]/(phi[j+1]-phi[j])) # find the distance r1 between x_{j+1} and
            the moving interface (x_{j+1} - s = r2*dx)
78      A = np.zeros((M,M), dtype = np.double)
79
80      T0n = T_analytical(lam, xmin, t) # temperature at left boundary at time t_n
81      T0n1 = T_analytical(lam, xmin, t+dt) # temperature at left boundary at time t_{
            n+1}
82      TMn = T_analytical(lam, xmax, t) # temperature at right boundary at time t_n
83      Tmn1 = T_analytical(lam, xmax, t+dt) # temperature at right boundary at time t_
            {n+1}
84
85      for i in range(M):
86          if i <j:
87              A[i,i] = 2+2*kl*dt/(dx**2)
88              A[i,i+1] = -kl*dt/(dx**2)
89              if i!=0: A[i,i-1] = -kl*dt/(dx**2)
90          elif i==j:
91              A[i,i] = 1+kl*dt/(dx**2) * 2/r1
92              A[i,i-1] = -kl*dt/(dx**2) * 2/(1+r1)
93          elif i==j+1:
94              A[i,i] = 1+ks*dt/(dx**2)* 2/r2
95              A[i,i+1] = -ks*dt/(dx**2) * 2/(1+r2)
96          elif i>j+1:
97              A[i,i] = 2+2*ks*dt/(dx**2)
98              A[i,i-1] = -ks*dt/(dx**2)
99              if i!=M-1: A[i,i+1] = -ks*dt/(dx**2)
100     A[0,0] = 2+3*kl*dt/(dx**2)
101     A[M-1,M-1] = 2+3*ks*dt/(dx**2)
102     Rn = np.zeros(M, dtype = np.double)
103     a1 = kl*dt/(dx**2)
104     a2 = ks*dt/(dx**2)
105     for i in range(1,j):
106         Rn[i] = a1*Ti[i-1] + (2-2*a1)*Ti[i] + a1*Ti[i+1]
```

```
107         for i in range(j+1,M-1):
108             Rn[i] = a2*Ti[i-1] + (2-2*a2)*Ti[i] + a2*Ti[i+1]
109         Rn[0] = (2-3*a1)*Ti[0] + a1*Ti[1] +2*a1*T0n +2*a1*T0n1
110         Rn[M-1] = a2*Ti[M-2] + (2-3*a2)*Ti[M-1] + 2*a2*TMn + 2*a2*Tmn1
111         Rn[j] = Ti[j]
112         Rn[j+1] = Ti[j+1]
113         Ti = np.linalg.solve(A,Rn)
114         return Ti
115     def pos_s(phi, xi):
116         """This function returns the position of the moving interface, using the values
                of the signed distance function"""
117         j = np.min(np.where(phi<0))-1 # find the index of the grid point x_j
118         r1 = -(phi[j]/(phi[j+1]-phi[j]))
119         xs = xi[j] + r1*dx # find position of the moving boundary
120         return xs
121
122     ### Run the program for the given numbers of time- and spatial steps
123     lam = fsolve(Transcendental, 1.0)[0] # compute the lambda necessary to find the
            analytical solution
124
125     for Mcount, M in enumerate(M_array): # go through al spatial numbers of steps
126         for acount, a in enumerate(a_array): # go through al a's
127             dx = dx_array[Mcount] # find dx
128             dt = a*dx**1 # find dt
129             print("dx =", dx)
130             print("dt =", dt)
131             N = round((tmax-tmin)/dt) # compute the number of time steps for which to
                    perform the calculation algorithm
132             print("Simulation for M =", M, "and a =", a, "so N =" , N)
133             print("dt/(dx**2) = ", dt/(dx**2))
134             xi = np.linspace(xmin+1/2*dx,xmax-1/2*dx, M, dtype = np.double) # set a
                    grid with xi at the center of each grid cell Ki
135
136             # Initalize t, phi, Ti
137             s = s_analytical(lam, tmin) # find the initial boundary position
138             t = tmin # set initial time
139             phi = s - xi # set initial signed distance function
140             Ti = T_analytical(lam, xi, tmin) # set initial temperature distribution
141
142             # Create arrays in which the solution is stored
143             tarray = np.array([tmin], dtype = np.double) # array with times
144             snum = np.array([s], dtype = np.double) # array with numerical boundary
                    position
145             sana = np.array([s], dtype = np.double) # array with analytical boundary
                    positions
146             Tnum = np.zeros((N+1, M), dtype = np.double) # set array with numerical
                    temperature
147             Tnum[0] = Ti
148             Tana = np.zeros((N+1, M), dtype = np.double) # set array with analytical
                    temperature
149             Tana[0] = Ti
150             vels = np.array([], dtype = np.double)
151
152             for timestep in range(N):
153                 v = find_vel(Ti, xi, phi, dx) ### Step 1: find the front velocity
154                 vels = np.append(vels, v) ### Store velocity in an array
155                 phi = solve_advection(vels, phi, M, dx, dt) ### Step 2: solve the
                        advection equation
156                 Ti = update_temp(phi, Ti, t, lam, M, dx, dt) ### Step 3: solve the heat
                        equation
157                 t = t+dt ### Step 4: update counter
158
159                 # Check if CFL-condition on advection equation is satisfied
160                 CFL = v*dt/dx
161                 max_CFL = 0
162                 if CFL>max_CFL: max_CFL=CFL
163                 if CFL>1/2: print("CFL is too big. It is given by:", CFL, "at t =", t)
164
165                 ### Solve solutions
166                 xs = pos_s(phi, xi)
```

```
167                 snum = np.append(snum, xs)
168                 sana = np.append(sana, s_analytical(lam, t))
169                 tarray = np.append(tarray, t)
170                 Tnum[timestep+1] = Ti
171                 Tana[timestep+1] = T_analytical(lam, xi, t)
172 
173             # Compute the errors
174             error_L2 = np.sqrt(1/len(snum))* linalg.norm(snum-sana, ord=2)
175             errors[Mcount, acount] = error_L2
176             print("The error for this simulation is given by", error_L2)
177 
178 ### Plot the numerical error as a function of the spatial stepsize
179 if len(M_array)>2:
180     for acount, a in enumerate(a_array):
181         plt.loglog(M_array, errors[:,acount], "o--", label= "a = " + str(a_array[
                acount]))
182     a1 = np.transpose(errors)[0][0]/(1/M_array[0])
183     a2 = np.transpose(errors)[0][0]/(1/M_array[0]**2)
184     a3 = np.transpose(errors)[0][0]/(1/M_array[0]**3)
185     plt.loglog(M_array, a1/M_array, "-", label="Ideal 1st order")
186     plt.loglog(M_array, a2/M_array**2,"-", label="Ideal 2nd order")
187     plt.loglog(M_array, a3/M_array**3, "-", label="Ideal 3rd order")
188     plt.xlabel(r"Number of spatial steps")
189     plt.ylabel("Numerical L2-error")
190     plt.legend(prop={'size': 10}, loc="best")
191     plt.title("Numerical error of the FVM-CN method as a function of spatial
            stepsize")
192     plt.show()
```