Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Ellisiv Sætherø Steen

A PDE-Based Strategy for Reconstructing Curves from Irregular and Unstructured Sampled Data

Master's thesis in Applied Physics and Mathematics
Supervisor: Anne Kværnø
June 2021

NTNU
Norwegian University of
Science and Technology

Ellisiv Sætherø Steen

A PDE-Based Strategy for Reconstructing Curves from Irregular and Unstructured Sampled Data

Master's thesis in Applied Physics and Mathematics
Supervisor: Anne Kværnø
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU

Kunnskap for en bedre verden

Abstract

We consider the problem of two-dimensional curve reconstruction from pointwise measurements from a PDE perspective. The PDE approach is an implicit level set approach that aims to be robust for noisy and irregular data sets. The theory presented in this thesis can be extended to three-dimensional surface reconstruction without difficulty. It can be applied in various applications where the aim is to reconstruct a curve or surface from a set of data measurements with few assumptions.

This thesis provides the theoretical background for deriving new level set models, and implementational details to solve the problems numerically. The models are tested on four data sets constructed to demonstrate their strengths and weaknesses. In addition, the test cases validate the theoretical analysis performed for all derived models. The numerical simulations also demonstrate how to adjust the model parameters to adapt to specific configurations of sample points. The resulting curves are smooth and approximate the data points nicely provided good parameter choices. However, the solutions are only macroscopically stationary, and the solutions for the noisy data sets tend to show a slight bias.

Sammendrag

Vi ser på todimensjonal rekonstruksjon av kurver fra et PDE-perspektiv. PDE-perspektivet er en implisitt nivåsett-tilnærming med formål om å håndtere irregulære datasett med støy. All teorien i denne oppgaven kan utvides til det tredimensjonale tilfellet uten vanskeligheter, hvis man vil bruke metoden til å rekonstruere overflater. Metoden har en rekke bruksområder der overflater eller kurver skal rekonstrueres fra måledata og få antakelser kan tas på forhånd.

Oppgaven vil ta for seg den teoretiske bakgrunnen for å utlede nye nivåsettmetoder, samt implementeringsdetaljer knyttet til numeriske løsninger. Modellene testes på fire datasett, konstruert for å vise styrker og svakheter ved modellene. Testene validerer de teoretiske resultatene og simuleringene demonstrerer hvordan de ulike modellene kan justeres for å tilpasses spesifikke datasett. Når parametrene er valgt riktig viser resultatene glatte kurver som tilpasser seg datapunktene godt. Likevel er løsningene bare makroskopisk stasjonære og løsningene for datasett med støy viser en viss strukturell skjevhet.

Preface

This master's thesis marks the end of my five years at the Norwegian University of Science and Technology and completes my degree in applied mathematics. The work has been done in close collaboration with my supervisor at the Department of Mathematical Sciences, Anne Kværnø, together with her colleague from the University of Lund, Claus Führer and their Ph.D students Samson Seifu and Alemayehu Adugna.

First of all, I want to thank Professor Anne Kværnø for taking me in on short notice and giving me an interesting topic. She has been a motivator, discussion partner, and a great supervisor this semester, and I could not ask for anyone better. I also want to thank the rest of the team working with me on this project. For your sense of humor, our enjoyable Friday meetings, and valuable feedback and discussions. It has been a pleasure working with you, and I wish you luck with your further work and collaboration.

Most importantly, I would like to thank my family, friends, and partner for supporting me and being patient with me when I have been focused on my studies—not only this semester but all my five years of study. I also want to give special thanks to my classmates, who have given me invaluable motivation and companionship.

Contents

Figures

Chapter 1

Introduction

Shape reconstruction from irregular, sampled data is a challenging problem since we often have little a priori knowledge about the original shape. Shape reconstruction is necessary when an object is only observed through pointwise measurements. This is so, when the object cannot be observed explicitly, for instance, in medical imaging and geology. In such cases, the measurements often contain noise, and we usually have no information concerning the connections between the data points.

There exist different approaches and methods for solving the shape reconstruction problem. One strategy is to first find reasonable connections in the data points by using, for instance, Veroni diagrams or Delaunay triangulations [1–3]. From this, we can construct a surface from the obtained net of connected data points using interpolation techniques. Finding the right connections is essential but could be challenging, particularly when the data contains noise. Gaussian Process Regression is another method, which does not find connections between individual data points but, builds surface segments from regressions on clusters of neighboring points. This method has previously been applied successfully, especially for noisy data sets [4].

This thesis investigates an implicit approach, namely a level set method. The level set method does not prioritize resources on pre-processing to find any patterns or connections in the data. Instead, it makes an initial guess without assumptions on the structure and gradually improves the solution with respect to pre-defined quality measures using a PDE formulation. Furthermore, the curve is not described parametrically, making it possible to form complex shapes and provides the topological flexibility to

handle splitting and merging of surfaces naturally.

Level set methods for tracking surfaces moving with curvature-dependent speed were first introduced by S. Osher and J. A. Sethian in 1988 [5]. Many have used this paper as a stepping stone and applied level set methods to a variety of physical applications like multi-phase flow [6, 7] and crystal growth [8], in addition to image applications like image enhancement, noise reduction [8, 9] and shape detection [8, 10, 11].

The underlying application that motivated surface reconstruction for our team was the estimation of bedrock topography. This is a shape reconstruction problem since samples are taken of the sediment thickness, and these samples are used to estimate the full shape of the bedrock. Because the sediment thickness can only be described implicitly through measurements, this is an example where few assumptions can be made about the data. This motivated the use of an implicit method.

To apply an implicit method, we were inspired by a paper from 2011 authored by A. Claisse and P. Frey [12]. The paper proposed the level set method to obtain a low-curvature surface approximating a set of data points. In other words, the surface moves with a velocity, dependent on the distance to the sampled points and the surface's curvature. The final solution is thus located close to the points and has low curvature.

This thesis is mainly a preliminary study on level set methods in general. We investigate how they can be used to formulate mathematical models to gradually deform a curve over time to approximate a set of points. The models we derive will approximate a curve to a set of data points in $\mathbb{R}^2$, simplifying the surface reconstruction problem of the bedrock topography to a one-dimensional curve reconstruction problem. Hence, we have detached ourselves from the underlying application in order to study the theory of level set methods and how to apply them to construct models with specific traits. The content of this thesis can thus be used as a general introduction to level set methods for shape reconstruction independent of the application.

The thesis is divided into six chapters:

Chapter 1 presents the level set method in a shape reconstruction context and motivates this thesis.

Chapter 2 introduces the theoretical background of the level set methods and the tools needed to formulate specific level set models.

Chapter 3 applies the theory and constructs models to reconstruct curves from sampled data.

Chapter 4 discusses the implementational aspect of applying the models and how to reproduce the results that will be presented.

Chapter 5 presents the obtained results from different test cases and shows how the models can be adapted to different configurations of points.

Chapter 6 concludes the thesis and suggests future work.

Chapter 2

Background Theory

This chapter presents some preliminaries on general level set methods. Specifically, we will show how to derive the level set method from a moving curve, and we will present some key features. In addition, we include some theory on distance functions which is relevant for level set methods in general but will prove especially useful for our application. Finally, we look at shape derivatives, or more specifically derivatives of domain integrals, and how they can be utilized to model the level set method for a specific application.

2.1 Level Set Methods

Before we begin, we must clarify that all derivations in this section will be performed for curves in $\mathbb{R}^2$, but everything can be extended to surfaces in a multidimensional space $\mathbb{R}^n$. This simplification is done to be consistent with the modeling and implementation, which is only applied to curves in a two-dimensional space.

The level set formulation is an implicit representation of a closed curve or curves. The curve is described by being a constant value on some higher dimensional function. We explain this concept through a familiar example, namely level curves on a map. Provided a continuous ground surface elevation, the level curves, or contours, join points on the surface of equal elevation. All curves representing the same value, or elevation in the cartographic setting, are called *iso-curves* or *iso-contours*. On a map, there are

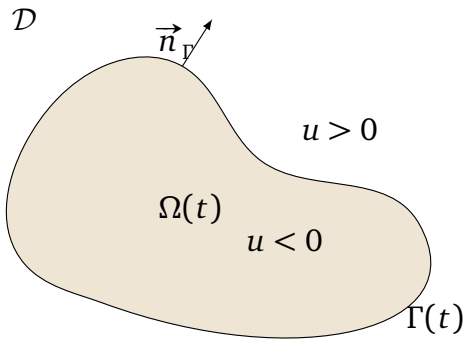


Figure 2.1: Level set representation of a curve $\Gamma(t)$ with outward pointing normal $\vec{n}_\Gamma$ located in a domain, $\mathcal{D}$. u is the higher dimensional level set function satisfying (2.1)-(2.3).

numerous iso-curves of different values separated by a constant height. Hence, the iso-curves effectively describe the steepness and height, and consequently, the shape of the ground in the area. In a level set context, we are not interested in the shape of the underlying higher dimensional function. We look at a single iso-value, which by standard practice is chosen to be the zero iso-value, which yields a curve called the *zero iso-contour*. The corresponding zero iso-contour(s) splits the domain into regions of two types: one where the underlying function is positive and one where it is negative. Level set methods track the shape of these curves and how they change over time.

2.1.1 Implicit Derivation

Now that we have introduced iso-curves, we can discuss the principle of the level set method. The goal is to represent a closed curve, $\Gamma(t)$, that moves under the influence of a velocity field that can change over time, $\vec{v}(\mathbf{x}, t)$. From now on, we use boldface letters for vectors and vector arrows for vector fields. The level set approach to this problem, as first presented in 1988 by S. Osher and J. Sethian [5], is to define a continuous function $u(\mathbf{x}, t)$ on a domain $\mathcal{D} \in \mathbb{R}^2$ containing the initial curve $\Gamma|_{t=0}$. The domain $\mathcal{D}$ is split into two parts by the curve $\Gamma(t)$, the interior Ω, and the exterior $\mathcal{D} \setminus \Omega$. See Figure 2.1 as a reference. The function $u(\mathbf{x}, t)$ must be constructed

in a way that satisfies the following properties

$$u(\mathbf{x},t) < 0 \qquad \text{for } \mathbf{x} \in \Omega(t), \tag{2.1}$$
$$u(\mathbf{x},t) = 0 \qquad \text{for } \mathbf{x} \in \Gamma(t), \tag{2.2}$$
$$u(\mathbf{x},t) > 0 \qquad \text{for } \mathbf{x} \in \mathcal{D} \setminus \Omega(t). \tag{2.3}$$

Now, the curve, $\Gamma(t)$, can be described in terms of $u(\mathbf{x},t)$ by being its zero iso-contour. Hence, if we can find the proper evolution of $u(\mathbf{x},t)$, we can implicitly track the motion of the curve. This means that we must find a model for the change in $u(\mathbf{x},t)$ to simulate the level curve, $\Gamma(t)$, as a curve flowing in the velocity field, $\vec{v}(\mathbf{x},t)$. Because the zero iso-curve of $u(\mathbf{x},t)$ is the only region of interest, we differentiate (2.2) with respect to time. Assuming that $u(\mathbf{x},t)$ is at least in $C^1(\Gamma(t))$, the following must hold.

$$\left(u(\mathbf{x},t)_t + \nabla u(\mathbf{x},t)\frac{\partial \mathbf{x}}{\partial t}\right)_{\mathbf{x}\in\Gamma(t)} = 0. \tag{2.4}$$

The term $\mathbf{x}_t = \frac{\partial \mathbf{x}}{\partial t}$ for $\mathbf{x} \in \Gamma(t)$ is the velocity of the curve. The curve has no density because it only represents a shape or boundary of a domain. Hence, the only component of the velocity that influences the movement is the normal component. We define $\vec{n}(\mathbf{x})$ to be the vector field pointing in the outward normal direction for all level curves and the positive direction of speed to be inwards. Hence, the velocity for the zero level curve $\mathbf{x}_t$ must be

$$\mathbf{x}_t = (\vec{v}(\mathbf{x},t)\cdot\vec{n})\vec{n} = -v_n\vec{n}, \quad \text{for } \mathbf{x}\in\Gamma(t). \tag{2.5}$$

We can also see from (2.1)-(2.3) that the gradient of $u(\mathbf{x},t)$ at the curve $\Gamma(t)$ is always pointing in the direction of the normal vector of $\Gamma(t)$, $\vec{n}$. Thus we can write the normal vector for all $\mathbf{x} \in \mathcal{D}$ as

$$\vec{n}(\mathbf{x}) = \frac{\nabla u(\mathbf{x},t)}{|\nabla u(\mathbf{x},t)|}. \tag{2.6}$$

Inserting (2.5) and (2.6) into (2.4), yields

$$\left(u(\mathbf{x},t)_t - v_n|\nabla u(\mathbf{x},t)|\right)_{\mathbf{x}\in\Gamma(t)} = 0,$$

which extended to the entire domain is the level set evolution equation.

The level set evolution equation

$$u_t - v_n|\nabla u| = 0 \tag{2.7}$$

We constructed this partial differential equation to let the curve, $\Gamma(t)$, flow in the velocity field, $\vec{v}(\mathbf{x}, t)$. We set no restrictions on the rest of the higher dimensional function, $u(\mathbf{x}, t)$. However, we see that we could have reproduced the calculations for any other iso-value of $u(\mathbf{x}, t)$ by defining

$$\begin{aligned} u(\mathbf{x}, t) &< k \qquad \text{for } \mathbf{x} \in \Omega_k(t), \\ u(\mathbf{x}, t) &= k \qquad \text{for } \mathbf{x} \in \Gamma_k(t), \\ u(\mathbf{x}, t) &> k \qquad \text{for } \mathbf{x} \in \mathcal{D} \setminus \Omega_k(t). \end{aligned}$$

and differentiating the iso-curve with respect to time. Since the right-hand side, k, is only a constant, this would make no difference to the resulting PDE. Consequently, we see that both the zero iso-contour and the entire function $u(\mathbf{x}, t)$ are transported in the velocity field $\vec{v}(\mathbf{x}, t)$.

We will now show that this implicit formulation yields the same solution as explicitly tracking the curve in the given velocity field. S. Osher and J. Sethian proposed the strategy in the paper [5], which introduced level set methods. Given a parametric curve with a specified speed, we can track its position through the equations of motion. These equations can be reformulated to yield an implicit formulation identical to the general level set equation.

2.1.2 Explicit Derivation

We begin with a closed initial curve, Γ_0, that moves along the normal direction with speed $v_n = v_n(\mathbf{x}, t)$ for $\mathbf{x} \in \Gamma$. Given a non-zero velocity, the curve evolves and we let $\Gamma(t)$ be the set of curves for $t \in [0, \infty]$. In an explicit formulation, every curve $\Gamma(t)$ can be parameterized by a variable $s \in [0, S]$. Let the parameterized position vectors be denoted $\mathcal{C}(s, t) = (x(s, t), y(s, t))$ for every curve in $\Gamma(t)$. Now, for a fixed $s = s^*$ this can be viewed as the Lagrangian perspective, following a certain particle moving with speed v_n. Fixing the time $t = t^*$ yields the parameterized curve $\mathcal{C}(s, t^*) = \Gamma(t^*)$.

The tangent vector field of the curve, $\vec{T}(\mathcal{C})$ comes easily from the parameterization by derivation with of x and y respect to s:

$$\vec{T}(\mathcal{C}) = \begin{bmatrix} x_s \\ y_s \end{bmatrix}.$$

Since the unit normal is orthogonal to the tangent, we write it as

$$\vec{n}(\mathcal{C}) = \frac{1}{\sqrt{x_s^2 + y_s^2}} \begin{bmatrix} y_s \\ -x_s \end{bmatrix}. \tag{2.8}$$

We now get the equations of motion for a curve moving with a known speed, v_n, in the normal direction,

$$\mathcal{C}(s,t)_t = \begin{bmatrix} x_t \\ y_t \end{bmatrix} = v_n \vec{n} = \frac{v_n}{\sqrt{x_s^2 + y_s^2}} \begin{bmatrix} y_s \\ -x_s \end{bmatrix}. \tag{2.9}$$

The function $\mathcal{C} : [0,S] \times [0,\infty) \to \mathbb{R}^2$ forms a continuous mapping from $(t,s) \to (x,y)$. The Jacobi matrix of this mapping is defined by the relations

$$\begin{aligned} dx &= x_s\, ds + x_t\, dt, \\ dy &= y_s\, ds + y_t\, dt. \end{aligned}$$

In matrix form, this is written as follows

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = J \cdot \begin{bmatrix} ds \\ dt \end{bmatrix} = \begin{bmatrix} x_s & x_t \\ y_s & y_t \end{bmatrix} \cdot \begin{bmatrix} ds \\ dt \end{bmatrix}. \tag{2.10}$$

The determinant of the Jacobi matrix, called the Jacobian, is

$$|J| = x_s y_t - x_t y_s = v_n \frac{-x_s^2 - y_s^2}{\sqrt{x_s^2 + y_s^2}} = -v_n \sqrt{x_s^2 + y_s^2}, \tag{2.11}$$

where we have used (2.9) for the relation between $(x_t, y_t) \to (x_s, y_s)$.

As long as the Jacobian is non-zero at a given point, the inverse function theorem [13] claims that there exists an inverse mapping locally around the point. That is the case here as long as the normal speed, v_n, is non-zero and the parameterization is chosen to avoid $x_s = y_s = 0$ simultaneously.

Hence, there locally exist an inverse mapping $\mathcal{C}^{-1} : (x,y) \to (t,s)$ and consequently a relation $t = f(x,y)$. An example of how the function f can look is displayed in Figure 2.2. Here we can see that this function is well defined if a curve does not cross the same spatial point more than once. If the speed function is continuous and non-zero, this is fulfilled. Then the function $f(x,y)$ can be used to implicitly describe the set of curves $\Gamma(t)$ by its iso-contours.

Using this, we can transform (2.9) into a PDE governing the motion of the curve, as shown in the following proposition.

Proposition 2.1. *The inverse mapping $t = f(x,y)$ must satisfy*

$$v_n^2 (f_x^2 + f_y^2) = 1, \tag{2.12}$$

for a non-zero $v_n \in C^0$.

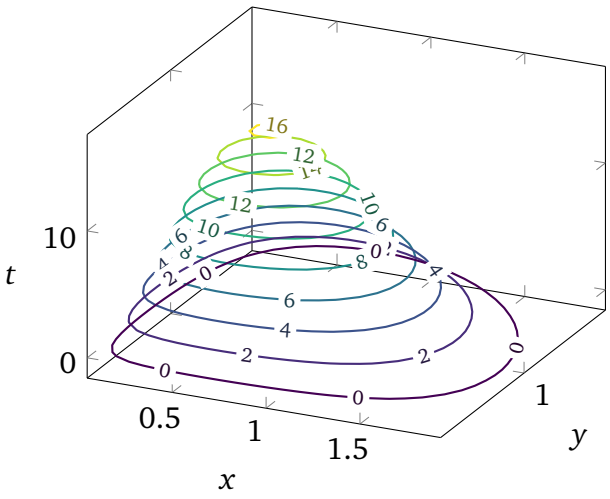


Figure 2.2: An example of the function $t = f(x, y)$ for some curves in the set $\Gamma(t)$. The contours of $f(x, y)$ represents the curves and the value of f represents the time at which the curves occurred.

Proof. The derivative dt, can be written in two ways.

$$dt = t_x dx + t_y dy, \tag{2.13}$$

$$dt = \frac{1}{|J|}(y_s dx - x_s dy). \tag{2.14}$$

The equation (2.13) is the total derivative, and (2.14) comes from (2.10) solved for dt. By comparing (2.13) and (2.14) we get that the terms in front of dx and dy have to be equal and thus

$$t_x = \frac{y_s}{|J|}, \qquad t_y = -\frac{x_s}{|J|}. \tag{2.15}$$

Moreover, using the relation between the Jacobian and the normal velocity from (2.11), we get

$$f_x^2 + f_y^2 = t_x^2 + t_y^2 = \frac{y_s^2 + x_s^2}{|J|^2} = \frac{1}{v_n^2}.$$

□

The partial differential equation (2.12) can be solved for (x, y, f), where $t = f(x, y)$ is the time the curve passed a spatial point (x, y). We can see from (2.15) that all the information needed to solve (2.12) is possible to

obtain from the given parameterization of the initial curve. Hence, we can find the entire three-dimensional surface described by $f(x,y) = t$ only from its boundary where $f(x,y) = 0$. We now prove that the solution of (2.12) yields the same solution as the level set evolution equation (2.7).

The trick to reformulate this into an implicit formulation, as we did for the implicit derivation, is to define a higher dimensional function $u(\mathbf{x},t)$ satisfying (2.1)-(2.3). The level curves of this function is defined by constant values of $u(\mathbf{x},t)$, and $t = f(x,y)$ as before. In two dimensions, $u = u(x,y,f(x,y))$ and since the level curves have constant values,

$$0 = \frac{\mathrm{d}u}{\mathrm{d}x} = u_x + u_t f_x,$$
$$0 = \frac{\mathrm{d}u}{\mathrm{d}y} = u_y + u_t f_y.$$

Solving for f_x and f_y, we directly get the relations

$$f_x = \frac{-u_x}{u_t}, \qquad f_y = \frac{-u_y}{u_t}. \tag{2.16}$$

Inserting (2.16) into (2.12), the PDE governing the motion of the level curves is $u_t^2 = v_n^2(u_x^2 + u_y^2)$ and taking the square root yields

$$u_t = \pm v_n(u_x^2 + u_y^2)^{1/2} = \pm v_n|\nabla u|,$$

where the sign decides the direction of propagation.

Because $u(\mathbf{x},t)$ is defined to be negative inside the curve, decreasing $u(\mathbf{x},t)$ means outward propagation of the zero level curve, $\Gamma(t)$. Look at Figure 2.3 for reference. Previously, we defined positive speed as inward-pointing. It follows that the higher dimensional function must increase, and we consequently choose the positive sign. Had either positive velocity been defined outwards, or the level set function been positive inside by definition, we would choose the negative sign. Hence, we see that the sign is only a question of definition.

The resulting equation is the level set evolution equation we found earlier:

$$u_t - v_n|\nabla u| = 0.$$

2.1.3 Distance- and Curvature-Dependent Speed

In the modeling aspect of this thesis, which will be discussed in Chapter 3, we want the level set velocity to transport a curve towards a set of sam-

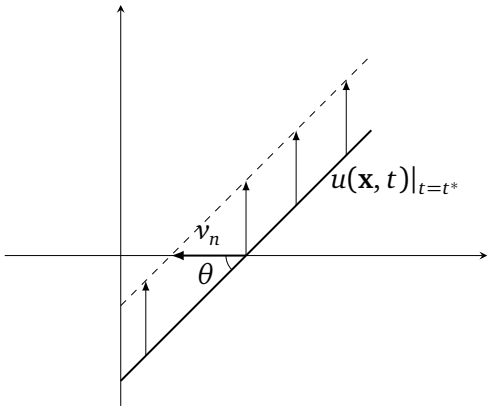


Figure 2.3: A cross section of a level set function $u(\mathbf{x},t)$satisfying (2.1)-(2.3). When $u(\mathbf{x},t)$ increases, the zero level curve moves with a speed v_n inwards.

pled points, $\mathcal{V}$. We define the point set to be $\mathcal{V} = \{\mathbf{v}_r\}$, $r = 1,2,\ldots R$, where $\mathbf{v}_r \in \mathbb{R}^2$ are coordinates of the sampled curve and R is the number of measurements taken. The normal velocity, v_n, from before, will for this case, both be dependent on the curvature, $\kappa(\Gamma(t))$, and a constant distance function $d(\mathbf{x};\mathcal{V})$. The distance function, $d(\mathbf{x};\mathcal{V})$, expresses the distance from any point $\mathbf{x} \in \mathcal{D}$ to the set of sampled points, $\mathcal{V}$. This function will be further discussed later, but for now, we only need to know that it is constant and well-defined on the entire domain.

We will now see that all the derivations above holds also for the distance- and curvature-dependent velocity function, $v_n = v_n(d(\mathbf{x};\mathcal{V})|_{\mathbf{x}\in\Gamma(t)}, \kappa(\Gamma(t)))$. The distance function, $d(\mathbf{x};\mathcal{V})$, is constant in time even though it is spatially dependent. The distance function on the curve $d(\mathbf{x};\mathcal{V})$ for $\mathbf{x} \in \Gamma(t)$ will therefore vary in time, but be independent on the curve's shape. It is thus on the same form as the velocity in the derivations above; $v_n(d(\mathbf{x};\mathcal{V})|_{\mathbf{x}\in\Gamma(t)}) = v_n(\mathbf{x},t)$.

The curvature is, however, dependent on the shape of the curve and we need to express it as a function of the parameterized curves $\mathcal{C}(x(s,t),y(s,t))$. Otherwise (2.12) will not be solvable given an initial curve $\mathcal{C}(x(s,0),y(s,0))$. The curvature is defined do be the divergence of the normal vector, $\vec{n}(\mathcal{C})$, [14] and using (2.8) it can be written as

$$\kappa(\mathcal{C}) = \nabla \cdot \vec{n}(\mathcal{C}) = \frac{y_{ss}x_s - x_{ss}y_s}{(x_s^2 + y_s^2)^{\frac{3}{2}}}. \tag{2.17}$$

We see that all terms in (2.17) comes from the parameterization. Consequently, the equation (2.12) can, given an initial parameterized curve, be

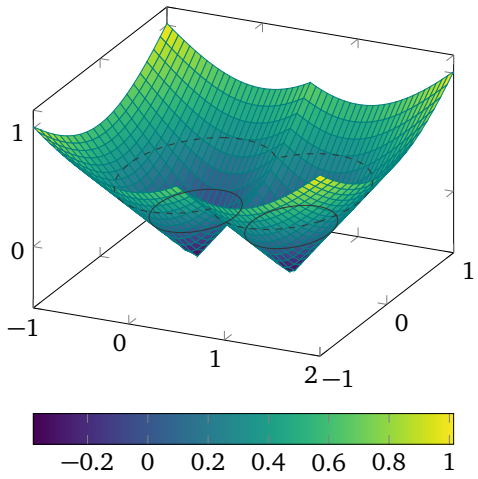


Figure 2.4: A three-dimensional function with two zero iso-contours plotted in solid lines and one contour with value of 0.4.

solved for (x, y, f) even with curvature- and distance-dependent speed.

It follows that the level set evolution equation is

$$u_t - v_n(\kappa(u), d(\mathbf{x}; \mathcal{V}))|\nabla u| = 0.$$

The curvature as a function of the higher dimensional function comes from taking the divergence of the normal vector of the level curves defined in (2.6), and is written

$$\kappa(u) = \frac{-u_{xx}u_y^2 + 2u_{xy}u_x u_y - u_{yy}u_x^2}{(u_x^2 + u_y^2)^{3/2}}. \tag{2.18}$$

We have now found a way to model a curve influenced by a velocity field that could be curvature- and distance-dependent. We conclude this general discussion of level set methods with some notes concerning the implicit formulation compared to an explicit tracking of the curve.

When the curve is described implicitly as a level curve, an advantage is that merging and splitting come naturally and do not need specific implementation. Take, for example, the function in Figure 2.4, which has two separate zero iso-contours, but only lowering the function by 0.4 merges the two contours into the single dashed iso-contour. So, merging of two separate iso-contours can be performed simply by lowering the higher dimensional function. Also, oppositely, if the shape of the higher dimensional function allows it, elevating the function could split a curve.

One drawback to the level set method is that instead of solving the equations of motion for the curve, we increase the complexity when we solve the general level set equation, (2.7), for the entire domain. It is only the curve that is relevant for the solution, and the shape and value of the higher dimensional function are irrelevant.

It is theoretically justified in [15] and [16] that the evolution of the curve is not dependent on the shape of the higher dimensional function as long as the zero level set is untouched. Calculations performed on the parts of the domain not containing the zero level set are thus wasteful. For this reason, it has been developed local level set methods that only solve the PDE in a layer around the curve. We refer to [17] for details and implementation of a local level set method.

2.2 Distance Functions

We have now presented the idea behind the general level set method. In the discussion, it was said that the shape of the higher-dimensional function, $u(\mathbf{x}, t)$, was insignificant to the shape and motion of the curve. In theory, this is true, but we will see that some properties are favorable, and as it turns out, a *signed distance function* is a natural choice of a higher dimensional function. This section will present the unsigned and signed distance function and justify why the signed distance is a natural level set function.

We begin with the unsigned distance function, which is everywhere the euclidean distance to an object. This is the standard distance function, but it is called *the unsigned distance function* in order to separate it from the signed distance function.

Definition 2.1 (distance function)**.** *[18] A distance function, applied to a point set, $\mathcal{V} = \{\mathbf{v}_1, \mathbf{v}_2, \dots \mathbf{v}_R\}$ for $\mathbf{v} \in \mathbb{R}^2$, yields the minimal euclidean distance from all spatial points $\mathbf{x} \in \mathbb{R}^2$ to the point set. Thus $d(\mathbf{x}; \mathcal{V})$ is defined as*

$$d(\mathbf{x}; \mathcal{V}) = \min_{\mathbf{v} \in \mathcal{V}} \|\mathbf{x} - \mathbf{v}\|_2. \tag{2.19}$$

When the function, d, measures the distance to a curve, Γ, it is denoted $d(\mathbf{x}; \Gamma)$ and defined as

$$d(\mathbf{x}; \Gamma) = \inf_{\mathbf{x}_\Gamma \in \Gamma} \|\mathbf{x} - \mathbf{x}_\Gamma\|_2. \tag{2.20}$$

The norm $\|\mathbf{x} - \mathbf{v}\|_2$ is positive for all input $\mathbf{x}$ and $\mathbf{v}$, and thus the distance function is globally positive, or *unsigned*.

The distance function can be computed to an arbitrary point set or curve, and the notion of a distance function makes intuitive sense. Solving the minimization problem can be time consuming, but as long as $\mathcal{V} \neq \emptyset$ or $\Gamma \neq \emptyset$, $d(\mathbf{x}, \cdot)$ is uniquely defined everywhere.

Furthermore, as long as there exists a well defined closest point, the gradient of an unsigned distance function $|\nabla d(\mathbf{x}; \cdot)| = 1$ everywhere ∇d is defined. However, the gradient ∇d is not defined where $\mathbf{x}$ is equidistant from at least two points in $\mathcal{V}$ or Γ, and when $d(\mathbf{x}; \cdot) = 0$ [18].

We denote the signed distance function as $u_d(\mathbf{x}; \Gamma)$. The function $u_d(\mathbf{x}; \Gamma)$ can only be applied to curves, or surfaces in general. It is a distance function that also provides information about whether or not a spatial point $\mathbf{x}$ is inside or outside the curve. It follows that we need information about where the inside and outside of the curve are to construct such function. Note that the inside can be established for a closed curve without prior knowledge, but this is not true for non-closed curves.

Definition 2.2 (signed distance function)**.** *In a domain, $\mathcal{D}$, including a closed region, Ω, with surface or boundary, Γ, the signed distance function $u_d(\mathbf{x}; \Gamma)$ is defined as*

$$u_d(\mathbf{x}; \Gamma) = \begin{cases} d(\mathbf{x}; \Gamma) & \text{if } \mathbf{x} \in \Omega, \\ -d(\mathbf{x}; \Gamma) & \text{if } \mathbf{x} \in \mathcal{D} \setminus \Omega. \end{cases}$$

The signs are exclusively a question of definition and could as easily be defined oppositely. What is convenient about this definition is that the signed

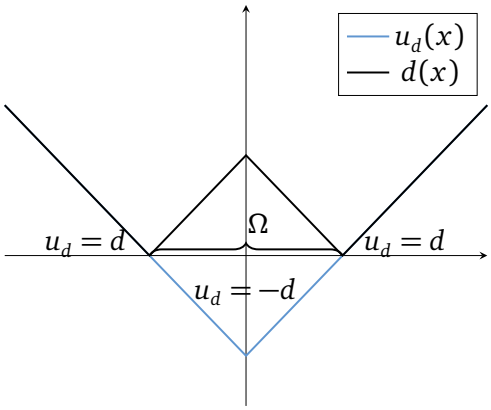


Figure 2.5: A cross section of the signed distance function, $u_d(x)$, and the (unsigned) distance function, $d(x)$, applied to a closed curve Ω.

distance function satisfies the conditions (2.1)-(2.3) concerning the higher dimensional level set function, $u(\mathbf{x}, t)$. A picture showing both the signed and unsigned distance function together can be viewed in Figure 2.5.

In addition to being qualified as a level set function, the signed distance function $u_d(\mathbf{x}; \Gamma)$ is a natural choice for the following reasons. First of all, it can be constructed easily from any initial curve, Γ_0, because signed distance functions are uniquely defined by their zero level set.

Since the distance function $d(\mathbf{x}, \Gamma)$ has the property of $|\nabla d| = 1$, the signed distance function $u_d(\mathbf{x}, \Gamma)$ inherits the same property, $|\nabla u_d| = 1$, through the definition. However, the gradient $\nabla u_d(\mathbf{x}; \Gamma)$ is defined also for $\mathbf{x} \in \Gamma$, which can be seen from Figure 2.5.

The property of the gradient is desirable in the level set equation (2.7) because the absolute value of the gradient decides the sensitivity of the higher dimensional function. The sensitivity can be seen in Figure 2.3 by observing that the gradient of the higher dimensional function decides the angle, θ, between the curve and the x-axis (or (x, y)-plane in $\mathbb{R}^2$). The inward velocity $v_n \sim \cos(\theta)$, which means that when the gradient increases, α increases and the velocity v_n decreases. Hence, if the level set function is a signed distance function, the sensitivity is constant over the domain, and a constant lifting leads to a constant inward velocity.

2.3 Gradient Flow and Derivatives of Domain Integrals

The motivation behind this section is to introduce *gradient flow*, an optimization strategy used to model the general level set equation (2.7) to a specific application. The gradient flow equation applied to the level set method relies on transformations produced by velocity fields and their corresponding derivatives. We will briefly go through the background for derivatives of domain integrals and we end this section with a useful result.

We ease into this subject by explaining the concept of gradient flow and its relation to the level set equation. We remember the level set equation $u_t - v_n|\nabla u| = 0$, where the normal speed, v_n, is extended from the desired speed of the zero level curve. We now want to derive this speed function to get a curve fulfilling the objective of this thesis; a curve with low curvature, approximating a set of sampled points. Hence, the velocity field should have a normal component that gradually deforms an initial curve until we reach a stationary situation where the objective is fulfilled. In order to construct the velocity function, we must mathematically define our objectives. By defining the desirable properties as measurable quantities, we can use optimization to find an optimal solution.

We approach the problem inspired by physics, introducing an energy functional measuring the potential energy of the curve through some properties we want to minimize. We use the notation from Section 2.1, where $\Omega(t)$ denotes the area bounded by the curve, $\Gamma(t)$. We define a general energy function, J, for a domain, $\Omega(t)$, bounded by the curve, $\Gamma(t)$, as

$$J(\Omega(t)) = \int_{\Omega(t)} f(\mathbf{x})\,\mathrm{d}\Omega + \int_{\partial\Omega(t)} g(\mathbf{x})\,\mathrm{d}S, \tag{2.21}$$

where f and g are the measurable quantities of the curve we want to control.

If we still follow the physical way of thinking, the natural state will minimize the potential energy field, and if not affected by other forces, this state will be stationary. The flow in a potential energy field will always move toward the fastest falling potential energy, and the same idea holds for gradient flow.

Gradient flow is a continuous version of the well known gradient descent method, also known as the steepest descent method. For an optimization

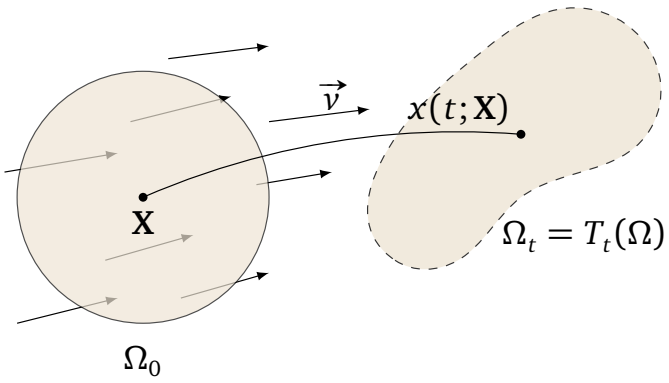


Figure 2.6: A transformation, T_t, of a domain $\Omega_0 \to \Omega_t$ by flowing in a velocity field $\vec{v}$ over a time t. The function $x(t;\mathbf{X})$ describes the path of a material point $\mathbf{X}$ moving in the velocity field.

problem on the form $\mathbf{x}^* = \arg\min_{\mathbf{x}} h(\mathbf{x})$, given an initial guess $\mathbf{x_0}$, we improve the solution by following the motion in negative gradient direction, $\mathbf{x}_t = -\nabla h(\mathbf{x})$.

We go back to the energy function, $J(\Omega)$. We want to decrease the energy functional by deforming the domain, which leads to a change in the integration domain, Ω. We have that, unlike $h(\mathbf{x})$, the input is not a coordinate in $\mathbb{R}^2$, but a domain of integration, $\Omega \subset \mathbb{R}^2$. In order to choose the optimal deformation to minimize the energy, we need to formulate how the energy changes when Ω is deformed. We define this change as the derivative of J and denote it $\mathrm{d}J(\Omega)$.

We assume that a domain, Ω, is a bounded, open set in $\mathbb{R}^2$ with a boundary $\Gamma = \partial\Omega$. A change in the domain into some Ω^t can be described by a transformation $T^t(\Omega) = \Omega^t$. The velocity method is about describing the domain as a continuum of points where all points are flowing in a velocity field, $\vec{v}$, which perturbs the shape of the domain. The transformation is thus driven by the velocity field. Look at Figure 2.6 for reference.

We consider a material point $\mathbf{X}$, moving under the influence of a velocity field, $\vec{v}(x,t)$. The trajectory of the material point, $\mathbf{X}$, in Eulerian coordinates is denoted $x(t;\mathbf{X})$, and will follow the velocity field, $\vec{v}(x,t)$. From this, we get the differential equation for the movement of the material points

$$\frac{\mathrm{d}x}{\mathrm{d}t}(\mathbf{X},t) = \vec{v}(x(t;\mathbf{X}),t), \quad x(t;\mathbf{X}) = \mathbf{X}, \quad t \geq 0.$$

The transformation, T^t, moves the material points along their trajectories given by the velocity field, which mathematically can be formulated as

$$\mathbf{X} \mapsto T^t(\mathbf{X}; \vec{v}) = x(t; \mathbf{X}).$$

For the domain, Ω, the transformation moves all material points in Ω along their respective trajectories. The time t in the transformation, T^t, represents how long we move along the trajectories. This means that $T^0(\Omega) = \Omega$, or in other words, $T^0 = I$. When $t \neq 0$, the total shape transformation of Ω along a velocity field, $\vec{v}$, is denoted as follows

$$\Omega^t(\vec{v}) = T^t(\vec{v})(\Omega) = \{T^t(\mathbf{X}; \vec{v}), \quad \forall\, \mathbf{X} \in \Omega\}.$$

We have now presented the notation we need to introduce derivatives of domain integrals. The following proposition and the proof can be found in a more general version in the book "Introduction to Shape Optimization" Section 2.31 and 2.33 by J. Sokolowski and J. Zolesio [19]. See also Lemma 2.1 in the paper by Claisse and Frey [12].

Proposition 2.2. *Let $\Omega(t)$ be a smooth domain in $\mathbb{R}^2$ bounded by the boundary curve $\Gamma(t)$. Define the functions $f(\mathbf{x}) \in W^{1,1}(\mathbb{R}^2)$ and $g(\mathbf{x}) \in W^{2,1}(\mathbb{R}^2)$ not dependent on the domain of integration, $\Omega(t)$. Define the functions J_1 and J_2:*

$$J_1(\Omega(t)) = \int_{\Omega(t)} f(\mathbf{x})\, \mathrm{d}\Omega,$$
$$J_2(\Omega(t)) = \int_{\Gamma(t)} g(\mathbf{x})\, \mathrm{d}S(\Gamma).$$

Let the integration domain, $\Omega(t)$, be transformed under the velocity field $\vec{v}(x,t) \in C^0$. The derivatives of J_1 and J_2 with respect to the integration domain at a fixed time $t = t^*$ is

$$\mathrm{d}J_1(\Omega(t^*), \vec{v}(t^*)) = \int_{\Gamma(t^*)} f(\mathbf{x})(\vec{v}(t^*) \cdot \vec{n})\, \mathrm{d}S(\Gamma), \tag{2.22}$$

$$\mathrm{d}J_2(\Omega(t^*), \vec{v}(t^*)) = \int_{\Gamma(t^*)} \Big(\frac{\partial}{\partial \vec{n}} g(\mathbf{x}) + \kappa(\mathbf{x}) g(\mathbf{x})\Big)(\vec{v}(t^*) \cdot \vec{n})\, \mathrm{d}S(\Gamma). \tag{2.23}$$

Now, returning to gradient flow, we need to find the direction of $\vec{v}(x,t)$ from Proposition 2.2 that maximizes $dJ(\Omega(t))$ and then move in the opposite direction. Since the gradient is dependent on the inner product $\vec{v}(t)\cdot\vec{n}$, the direction of maximal derivative is when $\vec{v}(t)$ is parallel to $\vec{n}$. For the general energy function, $J(\Omega)$, this means that the curve velocity Γ_t following gradient flow will have speed

$$\Gamma_t = -v_n\vec{n}_\Gamma = -(f(\mathbf{x}) + \kappa(\mathbf{x})g(\mathbf{x}) + \frac{\partial}{\partial\vec{n}}g(\mathbf{x}))\vec{n}_\Gamma. \tag{2.24}$$

Hence, we can find the optimal curve speed by defining the functions $f(\mathbf{x})$ and $g(\mathbf{x})$. This yields a flexible approach for modeling the general level set method, which will be useful in the following chapter.

Chapter 3

Modeling

We have now introduced the general level set method with curvature- and distance-dependent flow. However, we have not yet defined a velocity function to approximate the zero level curves to a set of points. Nevertheless, we have seen that if we can describe the desired properties of our curve through an energy function, we can move the curve in the steepest descent direction. As a result, the curve approaches an optimal curve for the defined properties using the gradient flow formulation. This chapter describes how to formulate reasonable energy functions and combine the gradient flow theory with the level set method. We use this to derive three specific models.

We saw in Section 2.3, that the optimal velocity function (2.24) is given by the functions $f(\mathbf{x})$ and $g(\mathbf{x})$. Defining these functions is the modeling aspect of the problem. In our case, the curve should approximate a set of points as close as possible while having low curvature. Consequently, we need to define the functions $f(\mathbf{x})$ and $g(\mathbf{x})$ to be measures of distance and curvature. In general, a minimal curve has the property of having zero mean curvature [20]. Thus we define $g(\mathbf{x}) = 1$. It follows that the gradient in the normal direction $\frac{\partial g}{\partial \vec{n}} = 0$.

The models we derive in this chapter have three different choices of distance-dependent velocity functions, which we denote $f_p(d(\mathbf{x}; \mathcal{V}))$, $p = 1, 2, 3$. We define a potential energy function, E, on the same form as (2.21) but including a weighting parameter $\alpha \in [0, 1]$ such that the smoothness of the

curves can be adjusted. We define it as

$$E(\Omega) = \alpha \int_{\Omega} f_p(d(\mathbf{x}; \mathcal{V}))\, \mathrm{d}\Omega + (1-\alpha) \int_{\partial\Omega} 1 dS. \tag{3.1}$$

We remember from the introduction to gradient flow that we want to minimize the energy function. We observe already now that if $f_p(d(\mathbf{x}; \mathcal{V})) > 0$ for all $\mathbf{x}$, this energy function is positive everywhere and the optimal curve is the trivial zero solution $\Gamma(t) = \partial\Omega = \emptyset$. We are not interested in the trivial solution, and we need to be aware of this when we construct appropriate distance measures $f_p(d(\mathbf{x}; \mathcal{V}))$.

Now, as presented in Section 2.3, we minimize the potential energy function (3.1) using gradient flow, and for that, we need to find the gradient, $\mathrm{d}E$, of E. We apply the differentiation formulas (2.22) and (2.23) from Section 2.3 for the two terms of E. The resulting derivative is

$$\mathrm{d}E(\Omega, \vec{v}) = \int_{\Gamma} \left(\alpha f_p(d(\mathbf{x}; \mathcal{V})) + (1-\alpha)\kappa(u)(\vec{v}(t) \cdot \vec{n}_{\Gamma}) \right) \mathrm{d}S.$$

Going in the direction of negative directional derivative means moving the curve with speed Γ_t defined in (2.24). Inserted for f and g, we obtain

$$v_n = \alpha f_p(d(\mathbf{x}; \mathcal{V})) + (1-\alpha)\kappa(u). \tag{3.2}$$

We get a level set model for a curve with minimal curvature and minimizing a function of the distance to a point set, by inserting the normal speed function (3.2) into the general level set method (2.7).

Generalized level set model

$$u_t = |\nabla u|(\alpha f_p(d(\mathbf{x}; \mathcal{V})) + (1-\alpha)\kappa(u)) \tag{3.3}$$

In the following, we will introduce the three distance-dependent functions, f_p, and the resulting models. Then, we will, for all models, perform some one-dimensional analysis to get a grasp of the theoretical behavior of the models.

3.1 Model 1

Model 1 is essentially the model proposed by Claisse and Frey [12]. They introduced a distance-dependent function that were linearly dependent on

an unsigned distance function to the point set. For this reason we begin by looking at a function

$$\hat{f}_1(d(\mathbf{x};\mathcal{V})) = d(\mathbf{x};\mathcal{V}).$$

As stated above, a distance-dependent function that is positive everywhere would yield no optimal curve satisfying $\mathrm{d}E(\Omega) = 0$, except for the trivial zero solution. Also, for the attraction term defined above, we see from (3.2) that even when the curve is inside the point set, the velocity would have direction inwards. A natural choice to avoid this is to define the distance to be negative inside the point set. This yields a curve moving outwards when inside the point set.

This brings us to an interesting question. How do we determine what is on the inside or outside of a set of points? When we discussed the signed distance function, the distance was related to a closed curve with a defined inside. When there are only points, we must somehow draw the border between the inside and outside. This border is a closed curve.

We denote the constructed closed curve as $\mathcal{C}_\mathcal{V}$, and the signed distance to that curve, $u_d(\mathbf{x};\mathcal{C}_\mathcal{V})$. Assuming that we can construct such a curve, we define the distance-dependent velocity as

$$\tilde{f}_1(d(\mathbf{x};\mathcal{V})) = u_d(\mathbf{x};\mathcal{C}_\mathcal{V}). \tag{3.4}$$

The signed distance function may seem like a nice solution. Outside $\mathcal{C}_\mathcal{V}$, the curve, $\Gamma(t)$, is pulled inwards by (3.4), and oppositely it is pulled outwards if inside $\mathcal{C}_\mathcal{V}$. With no curvature-dependent term, we would expect a final curve exactly equal $\mathcal{C}_\mathcal{V}$.

The problem is that we are looking for a curve that can approximate any set of points without assumptions on the structure of the sample points. Without information about the connections between the data points, we cannot construct, for instance, a polygon from the data points. This would otherwise have been a natural choice.

Hence, because (3.4) do not correspond well with the assumptions for the thesis, we approach the problem differently. With no information about the configuration of points, we go back to the unsigned distance function applied to our point set $d(\mathbf{x};\mathcal{V})$. At all points $\mathbf{x} \in \mathbb{R}^2$, $d(\mathbf{x};\mathcal{V})$ provides the distance to the closest sampled point in $\mathcal{V}$. Now the speed is decided by the distance to the closest point, and we want to construct a sign function $\sigma(\mathbf{x})$ to give the attraction the right direction. The distance function, $d(\mathbf{x};\mathcal{V})$, makes no assumptions on $\mathcal{V}$ and neither should σ.

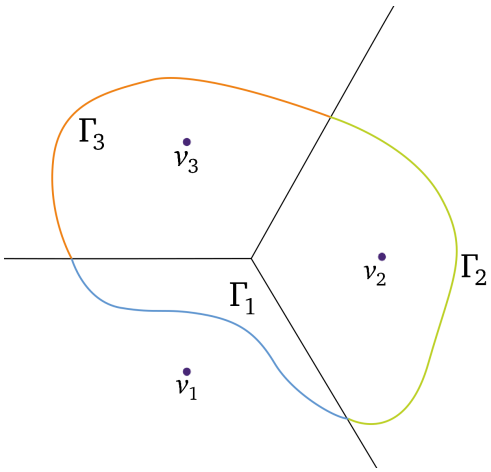


Figure 3.1: For a point set $\mathcal{V} = \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$, the lines divides the space into segments where $\mathbf{v}_r$ is the closest sample point. These lines divide the curve Γ into line segments Γ_r which should be drawn towards $\mathbf{v_r}$.

To construct the sign function, recognize that the only movement we are interested in is the movement of the zero iso-contour and the sign function only needs a reasonable sign at that curve. We can thus turn the problem around. Rather than checking if the curve is inside the point set, we detect whether or not the sampled points are outside the zero level curve. The curve, $\Gamma(t)$, is closed and must consequently have a meaningful inside and outside. In addition, by construction the sign of $u(\mathbf{x}, t)$ is negative inside $\Gamma(t)$ and oppositely positive outside as seen in (2.1)-(2.3). Hence, we can use the sign of $u(\mathbf{x}, t)$ to detect which side of the curve a sample point is on.

Using this, we construct the sign function, $\sigma(\mathbf{x}, t)$, for a fixed $t = t^*$ as follows. Look at Figure 3.1 for reference. Denote the rth sample point in $\mathcal{V}$ as $\mathbf{v}_r$, $r = 1, 2, \ldots R$. Divide the curve, Γ, into segments denoted Γ_r where $\mathbf{v}_r$ is the closest sample point for all $\mathbf{x} \in \Gamma_r$. Now, to move the curves to their closest point, we define a sign function fulfilling $\sigma(\Gamma(t^*)_r) = \mathtt{sgn}(u(\mathbf{v}_r, t^*))$. We can extend this function to all points in the domain by dividing $\mathcal{D}$ into sectors which in the implementation will be denoted as *the cohort* of $\mathbf{v}_r$. These sectors consist of all spatial points that has $\mathbf{v}_r$ as the closest sample point. The extended $\sigma(\mathbf{x}, t)$ is thus defined as

$$\sigma(\mathbf{x}, t) = \mathtt{sgn}((u(\mathbf{v}_r(\mathbf{x}), t))), \quad \mathbf{v}_r = \arg\min_{\mathbf{v} \in \mathcal{V}}(\|\mathbf{x} - \mathbf{v}\|_2). \tag{3.5}$$

Using the above, we get a distance-dependent speed, drawing the curve

towards the closest sample point at all times:

$$f_1(d(\mathbf{x};\mathcal{V})) = \sigma(\mathbf{x},t)d(\mathbf{x};\mathcal{V}). \tag{3.6}$$

The full model is obtained by inserting (3.6) into the general curvature-dependent model (3.3):

Model 1

$$u_t = |\nabla u|(\alpha\sigma(\mathbf{x},t)d(\mathbf{x};\mathcal{V}) + (1-\alpha)\kappa(x)), \quad \alpha \in \mathbb{R} \tag{3.7}$$

Remark: The way $\sigma(\mathbf{x},t)$ is defined in (3.5) violates the assumptions in Proposition 2.2 and we cannot find $\mathrm{d}E_1$ using the proposition. The sign function makes f discontinuous which makes the velocity discontinuous. This was not mentioned by Claisse and Frey [12] in their Lemma 2.1, which as we read it must have the same issue.

As we will see later, this does not damage the numerical results. The discretization makes the velocity discontinuous in any case. Thus, as long as the grid size is bounded away from zero, the discontinuity is not detectable. When the grid size is is bigger than some bounding $\varepsilon > 0$, one can construct a smooth function, $\tilde{\sigma}$, connecting $\sigma = 1$ and $\sigma = -1$ with a bounded derivative which is in L^1. The resulting velocity is continuous, and the discretization cannot separate σ from $\tilde{\sigma}$. However, this is not analyzed fully, since the discrepancy between the theory and the model was realized rather late in the process.

3.1.1 Radially Symmetric Analysis

We now reduce the situation down to a radially symmetric setting, and we perform some simplified analysis of the energy function. The setup can be viewed in Figure 3.2, where we have a circular curve $\Gamma(t)$ with radius $r_\Gamma(t)$ and center at the origin. The point set, $\mathcal{V}$, is also distributed circularly around the same center and with radius r_v. Furthermore, we assume that the density of the point set is so high that we can approximate the set of points as a continuous curve denoted $\Gamma_\mathcal{V}$.

Because of the symmetry and the high-density assumption, there are no spatial discontinuities because the entire curve is either inside or outside the point set. The resulting $\sigma(\mathbf{x},t) = \mathtt{sgn}\{r_\Gamma(t) - r_v\}$ is constant in space but still time-dependent. Now the energy function, E, is only a function of

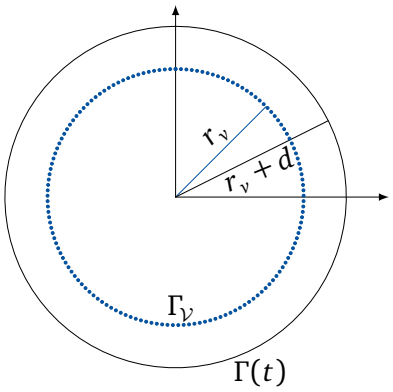


Figure 3.2: A radially symmetric set up with a point set, $\Gamma_{\mathcal{V}}$, and curve, $\Gamma(t)$ centered around the origin with radii independent of the angle with respect to the x-axis.

$r_\Gamma(t)$ and is written

$$E(r_\Gamma) = \mathtt{sgn}\{r_\Gamma - r_v\} \underbrace{\alpha \int_0^{r_\Gamma} (r - r_v) 2\pi r \, \mathrm{d}r}_{E_1} + \underbrace{(1-\alpha)\cdot 2\pi r_\Gamma}_{E_2},$$

$$E(r_\Gamma) = \begin{cases} 2\pi\alpha\left(\frac{r_\Gamma^3}{3} - \frac{r_\Gamma^2 r_v}{2}\right) + 2\pi(1-\alpha)r_\Gamma & \text{if } r_\Gamma \le r_v, \\ \alpha\left(\frac{2\pi r_v^3}{3} + 2\pi\left(\frac{r_\Gamma^3}{3} - \frac{r_\Gamma^2 r_v}{2}\right)\right) + (1-\alpha)2\pi r_\Gamma & \text{if } r_\Gamma > r_v. \end{cases} \tag{3.8}$$

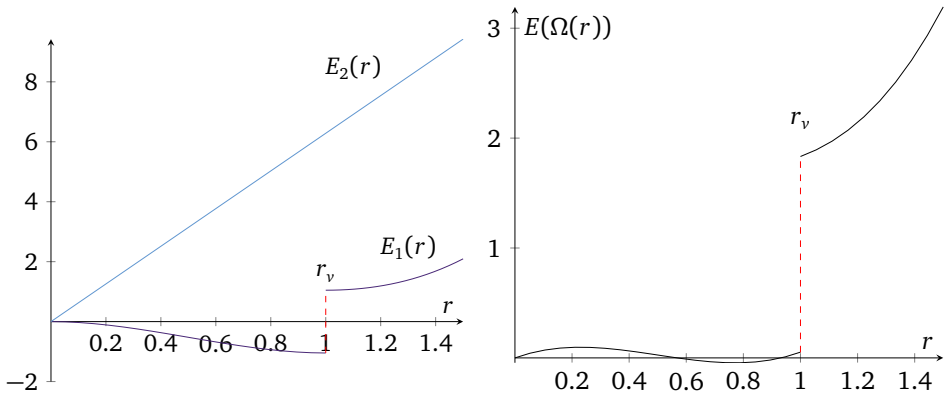
Note that if we introduce a physical unit for r, for instance, meters (m), we must add a scaling parameter to E_1 and E_2 to get a meaningful total energy. The total energy function is displayed in Figure 3.3b with its separate terms displayed in Figure 3.3a for specific parameters $\alpha = 0.85$ and $r_v = 1$. Here, we see that the energy function does obtain a minimum besides the trivial solution, and it is the global minimum. Note also that the minimum is not $r_\Gamma = r_v$, but inside the point set, where $r_\Gamma < r_v$.

Method of characteristics

A scalar hyperbolic conservation law is a PDE that can be written in the form

$$u_t + \nabla \cdot (f(u)) = 0, \tag{3.9}$$

where $f = (f_1, \ldots, f_m)$ and $x = (x_1, \ldots, x_n)$[21]. The model (3.7) is not a hyperbolic conservation law in the general two-dimensional case be-



(a) The separate energy terms **(b)** Total energy function

Figure 3.3: The potential energy function for model 1 (3.8) in the radially symmetric situation with $\alpha = 0.85$ and $r_v = 1$.

cause the curvature term is parabolic. However, in the one-dimensional case and the radially symmetric situation the model is hyperbolic. In the one-dimensional case, this is so because the curvature is zero, so we look instead further into the radially symmetric case.

The setup is the same as in the calculations leading up to (3.8), and is shown in Figure 3.2. For both $\mathcal{V}$ and Γ being circles with the same center, the distance $d(r;\mathcal{V}) = |r - r_v|$ and the curvature of a circle is known as

$$\kappa(r) = \frac{1}{r}. \tag{3.10}$$

The term $|\nabla u|$ can be expressed in terms of r through a simple change of variables:

$$\begin{aligned} r &= \sqrt{x^2 + y^2} \\ \implies \frac{dr}{dx} &= \frac{x}{\sqrt{x^2 + y^2}} \\ \implies \frac{dr}{dy} &= \frac{y}{\sqrt{x^2 + y^2}} \\ |\nabla u| = \sqrt{u_x^2 + u_y^2} &= \sqrt{\frac{\partial u}{\partial r}\frac{dr}{dx} + \frac{\partial u}{\partial r}\frac{dr}{dy}} \end{aligned}$$

$$\implies |\nabla u| = \sqrt{\frac{u_r^2}{x^2 + y^2}(x^2 + y^2)} = u_r, \qquad \text{if } u_r \geq 0. \tag{3.11}$$

Inserting (3.10) and (3.11) into the model equation in (3.7), we get

$$u_t = u_r v_n = u_r \left(\alpha\sigma(t)|r - r_v| + \frac{(1-\alpha)}{r} \right). \tag{3.12}$$

We now have a conservation law on the form (3.9) for r. Because the PDE is now only dependent on the radius, the total derivative of $u(r(t), t)$ with respect to time is

$$\frac{\mathrm{d}u}{\mathrm{d}t} = u_t + u_r\, r_t. \tag{3.13}$$

We want to investigate how the curve moves in time. The value of $u(\mathbf{x}, t)$ is by definition constant at the curve because it is the zero level curve. Moreover, the PDE can be simplified further at the curve since the sign function changes at the point when $r_\Gamma = r_v$. Hence, $\sigma(t)(|r_\Gamma - r_v|) = (r_\Gamma - r_v)$. The simplified PDE on the zero level curve is now

$$u_t = u_r \left(\alpha(r - r_v) + \frac{(1-\alpha)}{r} \right) \qquad \text{for } r = r_\Gamma(t). \tag{3.14}$$

Since $u(r(t), t)$ is constant at the level curves per definition, $\frac{\mathrm{d}u}{\mathrm{d}t} = 0$, so we set the left hand side of (3.13) equals zero and get

$$u_t = -u_r\, r_t. \tag{3.15}$$

By comparing (3.15) with (3.12), we see that

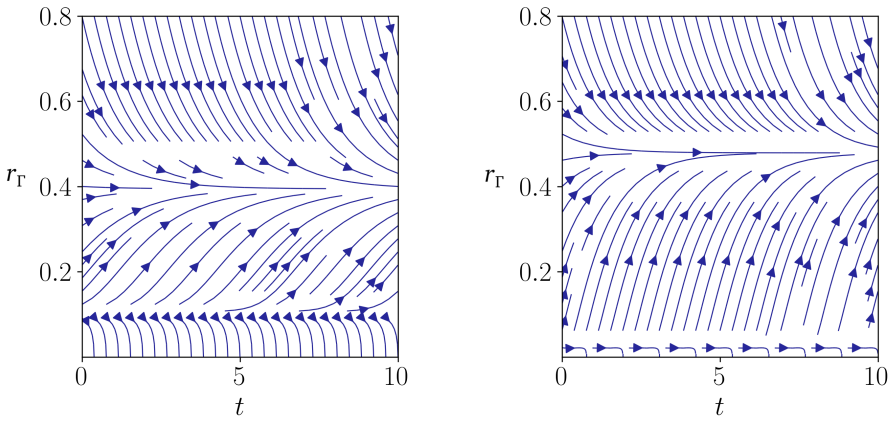
$$r_t = -v_n = -\left(\alpha\sigma|r(t) - r_v| + \frac{(1-\alpha)}{r(t)} \right), \tag{3.16}$$

where $r(t)$ is the radius for a single iso-contour which moves in time. We will call the trajectories of the curves following (3.16), the streamlines for the solution. First, we want to analyze how the zero level curve moves given an initial radius, r_0. We compare (3.14) and (3.15) to obtain the ODE for the streamline of a zero iso-contour

$$r_t = -\left(\alpha(r(t) - r_v) + \frac{(1-\alpha)}{r(t)} \right) \quad \text{for } r(t) = r_\Gamma. \tag{3.17}$$

The equation (3.17) is a separable equation, and the solution is an implicit function of $r(t)$,

$$\frac{\ln(\alpha(r(t)^2 - r_v\, r(t) - 1) + 1) - \dfrac{2\sqrt{\alpha}\, r_v \tan^{-1}\left(\frac{\sqrt{\alpha}(r_v - 2r(t))}{\sqrt{4 - \alpha(r_v^2 + 4)}} \right)}{\sqrt{4 - \alpha(r_v^2 + 4)}}}{2\alpha} = -t + C, \tag{3.18}$$



(a) $\alpha = 0.96$ **(b)** $\alpha = 0.99$

Figure 3.4: Streamlines for zero level set curves with initial radius r_0, following the ODE (3.17). The point set, $\Gamma_\mathcal{V}$ has radius $r_v = 0.5$.

where the constant, C, is decided from the initial conditions, $t = 0$, $r = r_0$ as follows

$$C = \frac{\ln(\alpha(r_0^2 - r_v r_0 - 1) + 1) - \frac{2\sqrt{\alpha}\, r_v \tan^{-1}\left(\frac{\sqrt{\alpha}(r_v - 2r_0)}{\sqrt{4-\alpha(r_v^2+4)}}\right)}{\sqrt{4-\alpha(r_v^2+4)}}}{2\alpha}. \tag{3.19}$$

We see the streamlines following (3.17) in Figure 3.4. It is important to notice that these are only streamlines for zero level curves given an initial radius, r_0.

We can also construct a general characteristic field for a situation with a fixed α, a point set radius r_v and initial radius, r_0 by solving (3.16). First we look at the term $\sigma(t)d(r)$ with a given radius of the zero level set curve r_Γ:

$$\sigma(t; r_\Gamma)d(r) = \quad |r - r_v| \qquad \text{when } r_\Gamma(t) \geq r_v, \tag{3.20}$$
$$\sigma(t; r_\Gamma)d(r) = -|r - r_v| \qquad \text{when } r_\Gamma(t) < r_v. \tag{3.21}$$

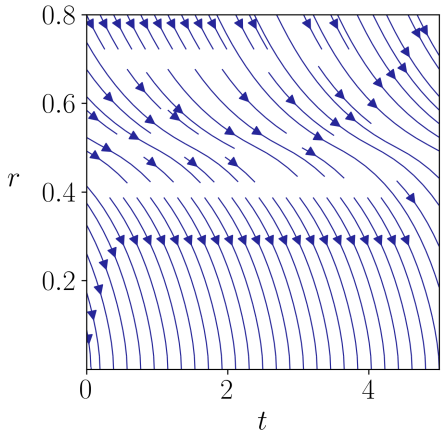
Using this, we can write (3.16) in terms of r_Γ as

$$r_t = -\left(\alpha|r(t) - r_v| + \frac{1-\alpha}{r(t)}\right) \qquad \text{when } r_\Gamma(t) \geq r_v, \tag{3.22}$$
$$r_t = \quad \left(\alpha|r(t) - r_v| - \frac{1-\alpha}{r(t)}\right) \qquad \text{when } r_\Gamma(t) < r_v. \tag{3.23}$$

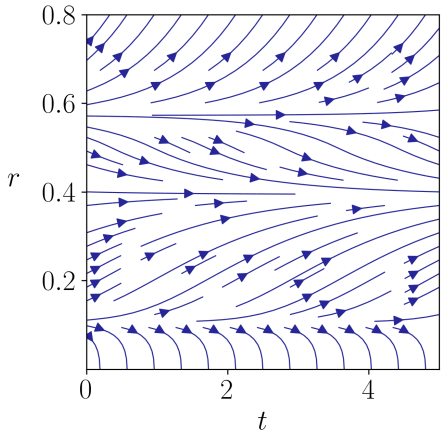
The only thing we need in order to make the full characteristic field is to find the time when $r_\Gamma = r_v$, which can be found from (3.18) and (3.19).

This is done in Figure 3.5 for a curve starting with $r_0 = 0.6$, with a point set, $\mathcal{V}$, situated in $r_v = 0.5$ and the weighting $\alpha = 0.96$. Figure 3.5 is the total characteristic field for all level curves. We see that the streamlines stemming from the area around the initial curve moves similarly to the zero iso-contours in Figure 3.4a, but further away, they differ more and more. That is because the sign change of $\sigma(r,t)$ happens more and more out of sync with when that particular level set falls into the point set. We see that even though all level curves flow in the velocity field given by v_n, the velocity is only constructed to approximate the zero iso-contour to the point set.

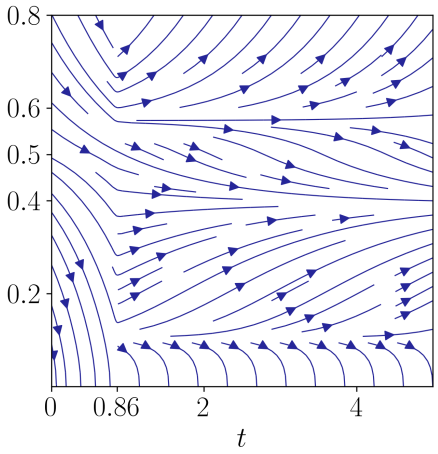
We also observe in the Figure 3.5 that level curves in a band around the zero level curve approaches the stationary radius as well. When numerous level curves approach the same radius, the higher dimensional function becomes steeper and steeper. This can cause problems for a numerical scheme which is sensitive to steep gradients.



(a) Streamlines with $\sigma(r,t) = 1$



(b) Streamlines when $\sigma(r,t) = -1$



(c) Streamlines for all level curves when the zero level set curve, $\Gamma(t)$ has initial radius, $r_0 = 0.6$.

Figure 3.5: The two figures on top shows streamlines when the sign function is $\sigma(r,t) = +1$ and $\sigma(r,t) = -1$, for a point set with radius, $r_\nu = 0.5$ and weight $\alpha = 0.96$. The lowermost figure shows the streamlines following (3.22) and (3.23), when the zero level set curve has initial radius $r_0 = 0.6$. The sign function $\sigma(r,t)$ changes sign when $r_\Gamma(t) = r_\nu$ which for this situation is in $t = 0.86$.

Stationary solutions

We now want to examine stationary solutions for the radially symmetric model in (3.12). In the context of level set methods, the stationary solution is obtained when the zero level curve is stationary. See for example in Figure 3.5 that when the zero level curve converges, the iso-curves above a certain value diverges and the entire function is never stationary. The stationary zero level curve can be obtained by finding the minima of $E(\Omega)$ or in the radially symmetric situation – finding $r_f = \lim_{t\to\infty} r(t)$ given an initial radius, r_0. In other words when $r_t = 0$ in (3.17).

We can see from Figure 3.4 that there are two stationary radii, r_f and that one is stable and the other is unstable. The stable solution is the outermost of two and is also the radius closest to the point set radius, r_v. We observe further by comparing the streamlines for the possible zero iso-curves for two different values of α, shown in Figure 3.4a and Figure 3.4b, that the radii of the stationary curves depends on α. We will now find the exact relation.

We set r_t equal zero in (3.17) and denote the radius where this is fulfilled as r_f. We thus obtain

$$0 = \alpha(r_f - r_v) + \frac{1-\alpha}{r_f}.$$

We multiply with r_f/α and obtain a quadratic equation

$$r_f^2 - r_v\, r_f + \frac{1-\alpha}{\alpha} = 0,$$

which has the solution

$$r_f = \frac{r_v}{2} \pm \frac{\sqrt{r_v^2 - 4(1-\alpha)/\alpha}}{2}. \tag{3.24}$$

First of all, we see that for $\alpha = 1$, $r_f = r_v$. Setting $\alpha = 1$ is equal to a velocity field that is only distance-dependent. As expected, the curve will cover the point set, unaffected by curvature. We also immediately see two solutions: two stationary radii, located at both sides of $r = r_v/2$. This is also what we observe in Figure 3.4. From before, we saw that only the outermost of the two solutions is stable, and the innermost is unstable. If the curve is located inside the innermost and unstable radius, it will shrink

to the zero-solution, $\Gamma = \emptyset$. We can from (3.24) further find the minimal α for which there exists any stationary solution.

When

$$r_v^2 < \frac{4(1-\alpha)}{\alpha},$$

there will be no real solution to (3.24). Solving for α gives the restriction for the weighting parameter, which is

$$\alpha \geq \frac{4}{r_v^2 + 4}. \tag{3.25}$$

When α does not fulfill (3.25), the zero level curve will have a velocity driven mainly by the curvature and with constant direction inwards, which will drive the radius smaller and smaller, until the circle disappears.

An important result is that when $\alpha < 1$, meaning whenever the curvature influences the motion, the stationary curve will always be inside the sampled points. The distance is given by $d(r_f) = |r_f - r_v|$ and inserting r_f from (3.24). We can see the same directly from our velocity function (3.2). When the curve covers our point set, the distance function is zero, but $\kappa(r) = 1/r_v \neq 0$. Hence, the curve still has speed inwards when $r_f = r_v$.

Thus we do not minimize both the curvature and distance separately, but we get a curve weighting the two. The result is a curve with low curvature close to the point set and high curvature further away.

3.2 Model 2

As seen from the analysis for the first model, the stationary solution for a circle of dense points would not approach the point set except for $\alpha = 1$. Further motivation for a new model came when we ran numerical results. For example, for three points forming an equilateral triangle, model 1 gave solutions as displayed in Figure 5.8. There, we see that the curve is flat at the sample points, and the total shape looks like an opposite triangle with corners far away from the sample points.

The new model aimed to approximate the equilateral triangle with smooth corners at the sample points for the same set of points. To generalize, this

means a model with high curvature close to the sample points and low curvature, approaching straight lines further away. This is similar to smoothed polygons with the point set as corners.

In order to do so, the distance-dependent function, $f_2(d(\mathbf{x};V)$, is chosen to be inversely proportional to the distance and is defined as

$$f_2(d(\mathbf{x};V) = \frac{\sigma(\mathbf{x},t)}{\beta d(\mathbf{x}) + \delta}. \tag{3.26}$$

Hence $f_2(d(\mathbf{x};V)$ is large close to the point set, yielding a stationary solution of high curvature. Furthermore, the distance-dependent term in the energy function acts similar to a gravitational field, attracting more and more the closer to the source. The resulting model is obtained by inserting (3.26) into the general level set model (3.3).

Model 2

$$u_t = |\nabla u|\left(\alpha\frac{\sigma(\mathbf{x},t)}{\beta d(\mathbf{x}) + \delta} + (1-\alpha)\kappa(u)\right), \qquad \alpha,\beta,\delta \in \mathbb{R} \tag{3.27}$$

The parameter $\delta > 0$ avoids the velocity having a singularity at the data points, and the term $\beta > 0$ is a scaling parameter for the distance. Without varying the parameter β with the domain size, identical point sets of different scalings would yield different curves.

3.2.1 Radially Symmetric Analysis

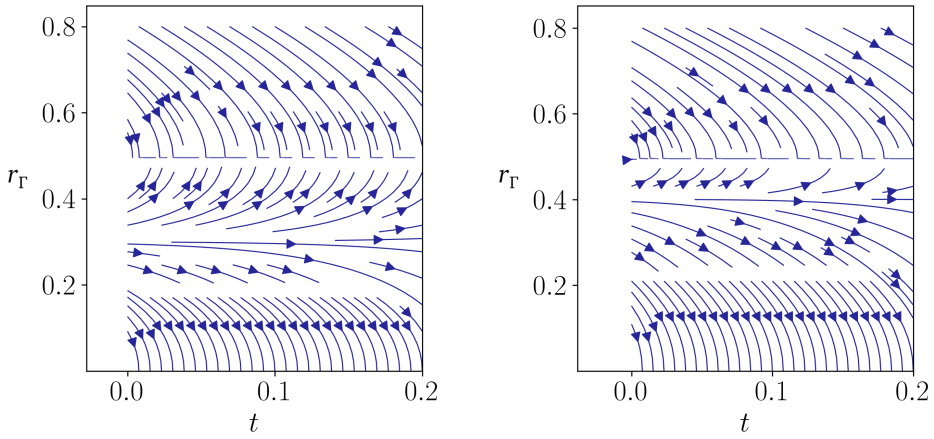
We conduct the same analysis for model 2 as for model 1. Using (3.10) and (3.11) we rewrite the curvature and gradient to be dependent of r. Using that $d(r) = |r - r_v|$, we obtain a radially dependent normal velocity

$$v_n = \frac{\alpha\sigma(r_\Gamma,t)}{\beta(|r-r_v|)+\delta} + \frac{1-\alpha}{r}. \tag{3.28}$$

Following the procedure for model 1 further, we find that the ODE for the characteristics/streamlines for the level curves is defined by $r_t = -v_n$. Using (3.20) and (3.21) to rewrite $\sigma(r,t)$ and by inserting everything into (3.28), we get the velocity for all level curves with a radius $r(t)$:

$$r_t = -\left(\frac{\alpha}{\beta(|r(t)-r_v|)+\delta} + \frac{1-\alpha}{r(t)}\right) \qquad \text{if } r_\Gamma \geq r_v, \tag{3.29}$$

$$r_t = \left(\frac{\alpha}{\beta(|r(t)-r_v|)+\delta} - \frac{1-\alpha}{r(t)}\right) \qquad \text{if } r_\Gamma < r_v. \tag{3.30}$$



(a) $\alpha = 0.4$ **(b)** $\alpha = 0.2$

Figure 3.6: Streamlines for the zero level set curve, $\Gamma(t)$, in the radially symmetric situation with the point set, $\mathcal{V}$, situated at $r_v = 0.5$ and for the velocity field depending on the inverse distance (3.28). The remaining model parameters are $\beta = 1$ and $\delta = 10^{-2}$.

We saw for model 1 that the ODEs for the streamlines could be simplified down to one equation for the zero level curves. Because the small constant δ is added in the denominator in (3.28), we cannot do the same for model 2. The equations (3.29) and (3.30) for the zero level curve with $r(t) = r_\Gamma(t)$, is given by the two equations:

$$r_t = -\left(\frac{\alpha}{\beta(r_\Gamma(t) - r_v) + \delta} + \frac{(1-\alpha)}{r_\Gamma(t)}\right) \quad \text{for } r_\Gamma(t) > r_v, \tag{3.31}$$

$$r_t = \left(\frac{\alpha}{\beta(r_v - r_\Gamma(t)) + \delta} - \frac{(1-\alpha)}{r_\Gamma(t)}\right) \quad \text{for } r_\Gamma(t) > r_v, \tag{3.32}$$

which correspond to the two cases where the curve is, respectively, outside and inside the point set. The streamlines for the zero level curves are displayed in a streamline plot in Figure 3.6 for two choices of α. It seems like there are two stationary radii, one at the point set, which is stable, and one unstable radius inside the point set.

We investigate the stationary solutions further by solving (3.31) and (3.32) with $r_t = 0$. The radii fulfilling this must be stationary and is denoted r_f. We begin with (3.31) which yields

$$\begin{cases} r_f = \frac{\beta(1-\alpha)}{\alpha+\beta(1-\alpha)}(r_v - \delta/\beta), \\ r_f > r_v. \end{cases}$$

We see that for all choices of $\alpha \in [0,1]$, $\delta > 0$ and $\beta > 0$; r_f will have to be smaller than r_v. This does not satisfy the constraint $r_f > r_v$, and thus the stationary solution of (3.31) is not feasible. We proceed to look at the situation where $r_\Gamma < r_v$ in (3.32) which has the stationary solution

$$\begin{cases} r_f = \frac{\beta(1-\alpha)}{\alpha+\beta(1-\alpha)}(r_v + \delta/\beta), \\ r_f < r_v. \end{cases} \tag{3.33}$$

For the values $\alpha = 0.4$, $\beta = 1$ and $\delta = 0.01$ as in Figure 3.6, we get $r_f = 0.306$ which is the inside $r_v = 0.5$. This must be the unstable stationary radius. What appeared to be a stable stationary solution in $r_\Gamma(t) = r_v$ is actually a discontinuity. We see this by inserting $r_\Gamma(t) = r_v$:

$$r_t = -\left(\frac{\alpha}{\delta} + \frac{(1-\alpha)}{r_v}\right) \quad \text{for } r(t) = r_\Gamma \geq r_v, \tag{3.34}$$

$$r_t = \left(\frac{\alpha}{\delta} - \frac{(1-\alpha)}{r_v}\right) \quad \text{for } r(t) = r_\Gamma < r_v, \tag{3.35}$$

which is non-zero for $\delta \neq (\alpha r_v)/(1-\alpha)$. The velocity is very high in opposite directions, slightly inwards or outwards of the point set radius, r_v. Thus, if we solve this equation numerically, we could observe oscillations around the radius of the point set if we use a fixed step size for the numerical time integration.

This oscillating behavior is important to notice. As we now have seen, there is no guarantee that we obtain a stable stationary solution for all configurations of point sets. Because the sign function is discontinuous and time-dependent, we can observe temporal discontinuities in the velocity function. However, since the sign draws the curve in the direction of the point set, it will still stay at $r_\Gamma(t) = r_v$, and the oscillations will depend on the size of the time steps.

Furthermore, the oscillations mean that we cannot make a similar streamline picture for all level curves for model 2 as we did in Figure 3.5 for model 1. For model 1, we got that the sign only changed once and the time of the sign change could be calculated, but for model 2, the sign will change infinitely fast and infinitely many times.

We thus do not get as much information about model 2 for this radially symmetric situation. Nevertheless, this is an interesting result because we have seen that not all point sets for all models yield a stable stationary solution. We have here seen that the zero level curve will stop at the point set for the provided example, but the requirement that $r_t = 0$ is not fulfilled.

3.3 Model 3

As we just saw, model 2 is constructed to have high curvature close to the sampled points only bounded by the small parameter $\delta > 0$. The idea for the third model, proposed by Seifu [22], is to still have an inverse relation to the distance but to bound the distance-dependent function in a more controlled way. The solution was the arctangent of the inverse distance:

$$f_3(d(\mathbf{x};V)) = \frac{\sigma(\mathbf{x},t)}{\pi/2}\arctan(1/(\beta d(\mathbf{x};V)))$$

When $d \to 0$, $f_3(d) \to 1$, and there is no need for the parameter δ to avoid a singularity. The resulting model is

Model 3

$$u_t = |\nabla u|\left[\frac{\alpha\sigma(\mathbf{x},t)}{\pi/2}\tan^{-1}\left(\frac{1}{\beta d(\mathbf{x};\mathcal{V})}\right) + (1-\alpha)\kappa(u)\right], \quad \alpha,\beta \in \mathbb{R} \tag{3.36}$$

Again, we do the same analysis on the radially symmetric example to see how the model moves the curve, $\Gamma(t)$.

3.3.1 Radially Symmetric Analysis

We proceed in the in the same manner as for the two previous models. First, we rewrite the equation from $u(x,y) \to u(r)$, and get

$$u_t = u_r\left(\frac{\alpha\sigma(r,t)}{\pi/2}\tan^{-1}\left(\frac{1}{\beta|r-r_v|}\right) + \frac{1-\alpha}{r}\right). \tag{3.37}$$

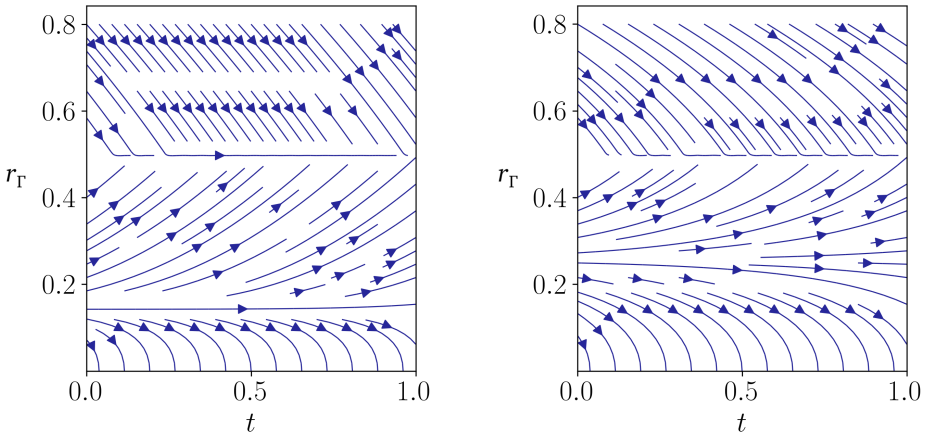
We can further derive the ODE for the streamlines for the zero level curve, $\Gamma(t)$, following the procedure presented for models 1 and 2, and obtain

$$r_t = \frac{\alpha}{\pi/2}\tan^{-1}\left(\frac{1}{\beta(r(t)-r_v)}\right) + \frac{1-\alpha}{r(t)} \qquad \text{for } r(t) = r_\Gamma(t). \tag{3.38}$$

Using (3.20) and (3.21) for the sign function $\sigma(r,t)$ in (3.37), we get the ODEs for all level curves:

$$r_t = -(\alpha\tan^{-1}\left(\frac{1}{\beta|r(t)-r_v|}\right) + \frac{1-\alpha}{r(t)}) \qquad \text{when } r_\Gamma(t) \geq r_v, \tag{3.39}$$

$$r_t = \quad (\alpha\tan^{-1}\left(\frac{1}{\beta|r(t)-r_v|}\right) - \frac{1-\alpha}{r(t)}) \qquad \text{when } r_\Gamma(t) < r_v. \tag{3.40}$$



(a) $\beta = 1$ **(b)** $\beta = 5$

Figure 3.7: Streamlines for the zero level set curves, $\Gamma(t)$, in the radially symmetric situation for model 3 defined in (3.36) with the point set, $\mathcal{V}$, situated at $r_v = 0.5$ and with parameter $\alpha = 0.9$.

Examples of the streamlines for the zero level curves can be viewed in Figure 3.7. Here we can see that compared to model 2, the velocity near the point set is not as big, but increasing the value of β yields a more similar shape for the streamline compared to model 2.

By looking at the streamlines in Figure 3.7 it seems like there are two stationary solutions. One unstable inside the point set and one stable at $r_\Gamma = r_v$. To check this, so we set r_t from (3.38) equal to zero. We denote the stationary radius as r_f and get

$$0 = \alpha \tan^{-1}\left(\frac{1}{\beta(r_f - r_v)}\right) + \frac{1-\alpha}{r_f}. \tag{3.41}$$

By inserting $r_f = r_v$, we can never fulfill (3.41) for r_v, $\alpha > 0$, since

$$0 \neq \alpha + \frac{1-\alpha}{r_v}.$$

Similar to model 2, r_f gives the unstable stationary curve and $r_\Gamma = r_v$ is a discontinuity.

Chapter 4

Implementation

We have looked at theoretical aspects of the level set method and constructed velocity models to reconstruct curves with minimal curvature from sampled data. The implementation and practical aspects have not been a subject. This chapter will answer questions concerning the technical implementation, the overall strategy chosen in this thesis, and possible pitfalls and solutions. The entire code base for the implementation is written in Python. However, most of this chapter explains logic and algorithms, which are adaptable to any preferred language.

More concrete, we will discuss the spatial and temporal discretization, suitable algorithms for producing the distance function and sign function, and how to prevent steep gradients from forming through a re-initialization procedure. We first present an outline of the main algorithm before going into detail to examine the different steps of the implementation.

Before we discuss this further, we remind ourselves of the three models we presented in Chapter 3. These are the partial differential equations we want our program to solve numerically.

$$u_t = |\nabla u|\,[\alpha\sigma(\mathbf{x},t)d(\mathbf{x};\mathcal{V}) + (1-\alpha)\kappa(u)], \tag{4.1}$$

$$u_t = |\nabla u|\left[\alpha\frac{\sigma(\mathbf{x},t)}{\beta d(\mathbf{x};\mathcal{V})+\delta} + (1-\alpha)\kappa(u)\right], \tag{4.2}$$

$$u_t = |\nabla u|\left[\frac{\alpha\sigma(\mathbf{x},t)}{\pi/2}\tan^{-1}\left(\frac{1}{\beta d(\mathbf{x};\mathcal{V})}\right) + (1-\alpha)\kappa(u)\right], \tag{4.3}$$

for $\alpha \in [0,1]$ and $\beta, \delta \in \mathbb{R}^+$.

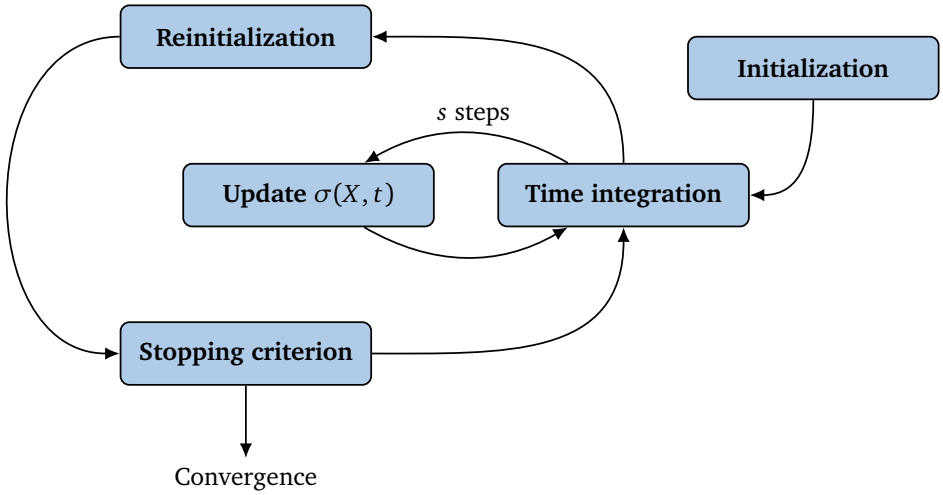


Figure 4.1: An overview of the main algorithm. We have an inner loop of s steps of time integration, before we re-initialize and check for convergence.

4.1 Outline of the Main Algorithm

An overview of the main steps in the implementation is found in Figure 4.1, where we see that our program consists of three phases. The first phase is the initialization, where we define all constants and initialize the grid, the curve, and the higher dimensional level set function. The complete initialization procedure will be presented in Section 4.2.

Then we have two loops, where we begin with the inner loop consisting of s steps of time integration followed by updates of the time-dependent sign function. The time integration procedure will be further explained in Section 4.3 and the sign update is described in Section 4.4. After s steps of time integration, the implementation moves to the third phase, which is the outer loop in Figure 4.1.

The outer loop consists of a re-initialization procedure and a convergence check. The higher dimensional function is perturbed, and the zero level curve consequently moved, through the time integration steps. Re-initialization in this context means substituting the perturbed level set function with a signed distance function to the zero level curve. We will go detailed through this in Section 4.5. The stopping criterion checks for a stationary curve and decides whether or not the program returns to phase two for further time integration.

4.2 Initialization

4.2.1 Constructing the Mesh

The initialization phase consists of defining all global constants and initial values for all variables. We start with the mesh used to discretize the spatial domain, $\mathcal{D}$. For this thesis, all the data sets, $\mathcal{V}$, are fabricated test cases, and the domain has been chosen as $\mathcal{D} = [-1, 1] \times [-1, 1]$. There several approaches for constructing a mesh to discretize the domain. When efficiency is essential, adaptive mesh strategies or local methods can be of use. However, the focus of this project was investigating the behaviors of the proposed models. Therefore, efficiency was not the priority, and we consequently chose a straightforward rectangular discretization.

The grid resolution is measured by the parameters N_x and N_y, the number of nodes in respectively x- and y- direction. The corresponding grid size is $h_x = 2/(N_x - 1)$ and $h_y = 2/(N_y - 1)$. The numbering follows the convention of NumPy's two-dimensional arrays. A grid point (x_j, y_i) is positioned in the ith row counted from the bottom and jth column counted from the left. We have used NumPy's `meshgrid` to produce the grid, which is represented by the matrix X containing both x- and y-coordinates. Note that this is a simplification of the notation, as `meshgrid` stores the mesh as X, Y, where the corresponding x- and y-variables are stored separately. A grid node in the mesh is denoted $X_{i,j} = (x_j, y_i)$.

4.2.2 Defining the Level Set Function

We denote the discretized level set function at a grid point $X_{i,j}$ and time t^n as $U^n_{i,j}$. In the initialization, we must define $U^0(X)$, which is needed for the first step of time integration. In the background theory for distance functions, presented in Section 2.2, we motivated why the signed distance function was a natural choice for $u(\mathbf{x}, t)$. It can be solely defined from a curve and has neither a steep or slow gradient.

An uncomplicated choice of the initial curve is the circle enclosing all sample points. This circle can be implemented without any prior assumptions about the distribution of points or information about the final solution. The generality fits our objective well. Also, this choice of the initial curve makes the construction of U^0 particularly easy. For a circular Γ^0 with radius r_Γ and

center in $\mathbf{c} = (c_1, c_2)$ we calculate the discretized signed distance function using Algorithm 1.

Algorithm 1: Signed Distance to a Circular Curve

Function `signed_distance(`X, r_Γ, $\mathbf{c}$`):`

```
    /* X: mesh, r_Γ: radius of initial curve, c: center of
     initial curve */
U^0 = zeros(N_y, N_x)
for i ← 0 to N_y do
    for j ← 0 to N_x do
        U[i,j] = ||X[i,j] − c||_2 − r_Γ
    end
end
return U^0
```

4.2.3 Initialization of the Distance and Sign Function

The point set, $\mathcal{V}$, is assumed to be a list of coordinates, $\{\mathbf{v}_r\} \in \mathbb{R}^2$, $r = 1, 2, \ldots R$ and R is the number of sample points. These coordinates are defined independent from the mesh, and hence, all algorithms needing information about $U(\mathbf{v}_r)$ become more challenging. In order to simplify some of the later procedures, we define an object containing the information we need. This means a slightly more comprehensive initialization, but it will make the following algorithms more intuitive.

It is natural to present the implementation of the distance function $d(\mathbf{x}; \mathcal{V})$ and the construction of the sample point object simultaneously. This is because one of the sample point's object variables can be computed within the initialization of the distance function. We come to this shortly, but first, we define the structure of the sample point object.

The object `sample_point` contains a state variable v.`sign`, the spatial coordinate of the sample point v.`coordinate`, the grid point to the lower-left v.`Xl` and its cohort v.`cohort`. The state variable is the sign of the higher dimensional function at the sample point, and as it turns out, the sign and the cohort of v will together be sufficient to update the sign function $\sigma(X, t^n)$. However, we will come to that in Section 4.4.

We define the cohort of a sample point, v_r, as all the grid nodes that has v_r as the solution to $\mathbf{v}_r = \arg\min_{\mathbf{v}\in\mathcal{V}}(\|\mathbf{x}-\mathbf{v}\|_2)$. Note that the object is denoted

v_r, while the coordinate has bald-face letter $\mathbf{v}_r$.

We initialize `sample_point` as shown in Algorithm 2. The initial sign is negative since the initial curve is encircling the point set by definition and all points lie on the inside of the curve. The coordinate is given, and from that information, we can also find the grid node placed on the lower left of the sample point. When we store the lower-left point, we always know which grid cell that contains the sample point. Note in Algorithm 2 that $(x_{\text{start}}, y_{\text{start}}) = X[0,0]$ which in our case means $(x_{\text{start}}, y_{\text{start}}) = (-1,-1)$. The cohort of v is still left to find.

Algorithm 2: Initialize Point Set

$v =$ **Object** `sample_point(v):`

```
    /* v: coordinate of sample point                    */
    v.sign = −1
    v.coordinate = v
    v.Xl = (floor((c[0]−x_start)/h_x), floor((c[1]−y_start)/h_y))
    v.cohort = None
```

The next step calculates the constant distance function from all spatial points to the point set and initializes the sign function. Since the initial curve encircles the point set, the sign function must be positive everywhere to transport the curve in the direction of the sample points.

The distance function is defined in (2.19). We must solve a minimization problem for all grid points: finding the closest sample point. Then, the distance function is defined as the distance to that point. We will see in Section 4.5 that there are more efficient procedures for calculating a distance function than Algorithm 3. However, there are two reasons why it is used in this setting. The first is that the initialization stage is not repeated, making it tolerable to lose some efficiency. Secondly, we utilize the procedure additionally to define the cohort of the sample points.

Algorithm 3: Initialize Distance and Sign Function

Procedure `init_distance_and_sign(`X, $\mathcal{V}$`):`

```
/* X: mesh, V: list of all sample points                    */
σ = ones(N_x, N_y)
d_v = zeros(N_x, N_y)
for i ← 0 to N_y do
    for j ← 0 to N_x do
        c_p = [‖X_{i,j} − v.coordinate‖_2 ∀ v ∈ V].argmin()
        d_v[i, j] = ‖X_{i,j} − c_p.coordinate‖
        c_p.cohort += ([i, j])
    end
end
return d_v, σ, V
```

This concludes the initialization procedure, and we are ready to advance to the second phase of the program.

4.3 Time Integration

The first step of the second phase is the time integration procedure. This step solves the differential equations numerically. We will now go through the discretization of the differential operators in the PDEs (4.1) – (4.3) and the temporal discretization technique. Generally, we can express all models as

$$u_t = |\nabla u|\,(\alpha f_p(\mathbf{x}) + (1-\alpha)\kappa(u)),$$

where $p = 1, 2, 3$, and f_p is the distance-dependent velocity for model p. Expanding all terms and using (2.18), this can be reformulated as

$$u_t = \alpha\,(u_x^2 + u_y^2)^{1/2} f_p(\mathbf{x}) + (1-\alpha)\,\frac{u_{xx}u_y^2 - 2u_{xy}u_x u_y + u_{yy}u_x^2}{u_x^2 + u_y^2}. \tag{4.4}$$

We use a central difference scheme for the spatial differential operators defined in (A.1) – (A.5). The implemented time integration scheme is the Explicit/Forward Euler method defined in (A.6) with a step size k. Forward Euler is very easy to implement, but it may require small step sizes to ensure stability. As expressed earlier, this implementation does not have the main focus on efficiency, and we only refer to J. Sethian [8] Chapter 6 for a more thorough review of suitable discretization techniques.

To solve the numerical system, we furthermore need to introduce boundary conditions. Because the curve is the only region of interest, the choice of boundary conditions is less important, and we choose standard Neumann boundaries. The following initial value problem is

$$\begin{aligned} u_t &= |\nabla u|\,(\alpha f_p(\mathbf{x}) + (1-\alpha)\kappa(u)), && \text{for } \mathbf{x} \in \mathcal{D} \\ \nabla u \cdot \vec{n}_{\mathcal{D}} &= 0, && \text{for } \mathbf{x} \in \partial\mathcal{D}, \end{aligned}$$

where $\vec{n}_{\mathcal{D}}$ is the outward-pointing normal to the boundary of the domain $\partial\mathcal{D}$.

We see from (4.4) that it is easy to switch between the different models by changing f_p. Flexibility is thus one of the advantages of this method; it is easy to formulate new models and quickly test their behaviors.

4.4 Updating the sign function, $\sigma(X, t)$

When we have performed a step of time integration, the curve Γ^n may have moved. If a curve section has passed a sample point, but the sample point is still the closest point for that curve section, the sign function must be updated. We have discussed the reasons why in Chapter 3, but we repeat it once more. Because the distance function is always positive, the sign function must give the attraction the right direction. We remember that positive velocity is defined inwards, so when the closest sample point is inside the curve, the sign must be negative, to pull it outwards.

The initialization procedure for the sign function was easy because the initial curve encircled the point set. We did not need the definition of the sign function at all because it was guaranteed to be positive on the entire domain. In the update procedure, on the other hand, we must use the definition to update only the right grid points. For that reason, we repeat the definition stated in (3.5), but now in its discretized version:

$$\sigma(X_{i,j}, t) = \mathtt{sgn}((u(\mathbf{v}_r(X_{i,j}), t))), \quad \mathbf{v}_r = \arg\min_{\mathbf{v}\in\mathcal{V}}(\|X_{i,j} - \mathbf{v}\|_2). \tag{4.5}$$

We see that solving (4.5) in practice consists of two parts: solving a minimization problem to find the closest sample point in the point set and calculating the sign of the higher dimensional level set function at that point. We will now, as we promised, use the information we get from the cohort of each sample point to skip the first step, namely solving the minimization problem.

Instead of recalculating (4.5) at every update, we only calculate the values at the sample points to see if an update is necessary. The cohort of a sample point contains all grid points that need updating if the sign of the sample point has changed. For a visual explanation of the cohort, go back to Figure 3.1, where the cohorts are the three regions dividing the domain. However, since the sample points are not on the grid, we cannot obtain the value of U on $\mathbf{v}_r$ directly. Instead, we need to estimate the value from the surrounding grid points by interpolation. We have implemented bilinear interpolation, which should be adequate since we have $|\nabla u| \sim 1$.

We can approximate the higher dimensional function $u(x, y, t)$ for a fixed $t = t^*$ on a grid cell $[x_i, x_{i+1}] \times [y_j, y_{j+1}]$ by bilinear interpolation using the following formula [23]

$$u(x,y) = \frac{1}{(x_{j+1}-x_j)(y_{i+1}-y_i)} \begin{bmatrix} x_{j+1}-x & x-x_j \end{bmatrix} \begin{bmatrix} U_{i,j} & U_{i,j+1} \\ U_{i+1,j} & U_{i+1,j+1} \end{bmatrix} \begin{bmatrix} y_{i+1}-y \\ y-y_i \end{bmatrix}. \tag{4.6}$$

We see that equation (4.6) needs the four neighboring grid points of each sample point. In the data structure we defined in Algorithm 2, we saw that the grid point to the lower-left was stored. Given the lower-left point, we can always find the grid cell containing the sample point. The four grid nodes of the cell can then be used to perform bilinear interpolation through equation (4.6). The algorithm is explained in detail in Algorithm 4

The complete algorithm for updating the sign function is to go through every sample point, estimate the sign of U at that point and check if it is equal to the state variable v.`sign`. If the sign has changed, the sign function at every grid point in the cohort of v changes sign. This is shown in Algorithm 5.

Algorithm 4: Linear Interpolation on Sample Point

Function `bilinear_interpolation`(X, U, v_r)**:**

```
/* X: mesh, U: discretized level set function, v_r:
   sample point                                          */
i_1, j_1 = v_r.Xl              // Grid numbers for closest point
x_1, y_1 = X[i_1, j_1]          // Grid values for closest point
x_v, y_v = v_r.coordinate                  // Coordinates of v_r
d_1 = x_v − x_1
d_2 = y_v − y_1
x_2 = x_1
y_2 = y_1
i_2 = i_1 + 1
j_2 = j_1 + 1
/* Define the four points needed for bilinear
   interpolation                                         */
U_1 = U[i_1, j_1]
U_2 = U_[i_1, j_2]
U_3 = U_[i_2, j_1]
U_4 = U_[i_2, j_2]
/* Solve (4.6)                                           */
b_1 = [h_x − abs(d_1), abs(d_1)]
A = [[U_1, U_2], [U_3, U_4]]
b_2 = [h_y − abs(d_2), abs(d_2)]
u_v = 1/(h_x h_y) b_1 A b_2^T
```

return u_v

Algorithm 5: Updating the Sign Function

Procedure `update_sigma`(X, $\mathcal{V}$, σ)**:**

```
/* X: mesh, V: list of sample points, σ: discretized
   sign function                                         */
for v ∈ V do
    u_v = bilinear_interpolation(v)
    if sgn(u_v) ≠ v.sign then
        σ[v.cohort] *= −1
        v.sign *= −1
    end
end
```

return $\sigma(\mathbf{x}, t^n)$

4.5 Re-Initialization

After s steps of time integration and updating the sign function, we advance to the outer loop in Figure 4.1. This stage is where we re-initialize and check a stopping criterion. This section will address the re-initialization and present some theory on calculating distance functions, which is needed in the procedure.

To start, we begin with the motivation for a re-initialization procedure. We discussed in Section 2.1 on level set methods that $|\nabla u|$ scales the change needed to move the curve, Γ, with a given speed, v_n. This means in practice that the curve is more sensitive to changes when the gradient is small, and oppositely, insensitive to changes if the gradient is large. It has also been shown in practice, for example, by D. Peng et al. [17] that the evolution of the curve may be unstable if the gradient is too steep or flat. Additionally, our discretization techniques are vulnerable to steep gradients, where oscillations could start to form.

Re-initialization aims to adjust the shape of the higher dimensional function to fulfill $|\nabla u| = 1$ while keeping the curve untouched. As we remember, the shape of the higher dimensional function does not impact the motion of the curve. Hence, we can perform this considerable change in $u(\mathbf{x}, t)$ without impacting curve evolution.

The strategy is to construct the higher dimensional function as a signed distance function to the current zero level curve. The function U^n is replaced with a function $\tilde{U} = u_d(X, \Gamma^n)$ using Definition 2.2 from Section 2.2. We note that in the implementation, we do not store the parameterization of the curve. The curve can be found as the zero iso-contour, for instance, using the `matplotlib.pyplot.contour()` function, which yields a list of intersection points where the level curve crosses the grid lines. Hence, numerically calculating the distance to a curve reduces to calculating the distance to a set of points.

Due to this, we can use the same strategy as in Algorithm 3 where we constructed $d(X, \mathcal{V})$ to make an unsigned distance function, and construct $\tilde{U}^n$ as

$$\tilde{U}^n = d(\mathbf{x}; \Gamma^n) \cdot \texttt{sign}(U^n). \tag{4.7}$$

However, this algorithm loops through all nodes and goes through a list of all points on the curve for each grid point. The size of the list Γ^n also scales linearly with the grid size, which culminates in a complexity of $\mathcal{O}((N_x \times$

$N_y) \times (N_x + N_y))$ which is far worse than $\mathcal{O}(N_x \times N_y \times \texttt{len}(\mathcal{V}))$ in Algorithm 3. In addition, the re-initialization is performed every sth time steps, which makes the high complexity far less acceptable.

We observed in the implementation that this strategy slowed down the runtime of the overall algorithm, and we were thus on the lookout for something else. The answer became the *Fast Marching Method* derived and developed by Sethian in the 90s. A thorough introduction can be found in his book "Level Set Methods and Fast Marching Methods" [8]. We provide a brief discussion of the Fast Marching Method below, which shows why the method is relevant to our situation. For the implementation, we only need to know that there exists a python package named `scikit.fmm`, which is easy to use and perfectly compatible with level set functions. Furthermore, it is open-source, and both the code and documentation can be found on GitHub [1].

Remark: The re-initialization does not necessarily need to be performed regularly at every sth step; it is also possible to check the gradients and set a criterion for how much they are allowed to deviate from $|\nabla U| = 1$.

4.5.1 The Fast Marching Method

The Fast Marching Method is a standard method for solving an equation called *the Eikonal equation*. We will first show that the distance function is only a special case of the Eikonal equation and then present the idea behind the Fast Marching Method.

Definition 4.1 (The Eikonal Equation)**.** *Assume that we are tracking a front* $\Gamma(t)$*, expanding with a velocity* $v_n(\mathbf{x}, t)$ *starting from an initial curve* $\Gamma(0)$*. The time of arrival,* $T(\mathbf{x})$ *is the time the front crosses the point* $\mathbf{x}$*. This function* $T(\mathbf{x})$ *can be found by solving the Eikonal equation [8]*

$$|\nabla T(\mathbf{x})| = \frac{1}{v_n(\mathbf{x}, t)}. \tag{4.8}$$

Recalling $|\nabla d(\mathbf{x}, \cdot)| = 1$, we immediately see that the distance function must satisfy (4.8) for a velocity $v_n = 1$. It is infact obvious that the travel time and distance traveled is identical when $v_n = 1$ by the physical fact that `time=distance/speed`. From now on, we replace the time of arrival T with the distance $d(\mathbf{x}, \Gamma)$ when $v_n = 1$.

[1] `https://github.com/scikit-fmm/scikit-fmm`

The time of arrival at a grid point $X_{i,j}$ is uniquely defined by the time of arrival of the neighboring grid points where the front has already crossed. Thus information spreads outwards, and upwind schemes are appropriate to use. The direction of the upwinding is decided by which direction the time increases. The resulting upwind scheme [8] for solving (4.8) for a speed of propagation $v_n = 1$ is

$$\begin{aligned}\Big[&\max(D_{i,j}^{-x} d(\mathbf{x},\Gamma),0)^2 + \min(D_{i,j}^{+x} d(\mathbf{x},\Gamma),0)^2 \\ &+ \max(D_{i,j}^{-y} d(\mathbf{x},\Gamma),0)^2 + \min(D_{i,j}^{+y} d(\mathbf{x},\Gamma),0)^2 \Big] = 1.\end{aligned} \tag{4.9}$$

The operators $D_{i,j}^{-x} d(\mathbf{x},\Gamma)$ and $D_{i,j}^{+x} d(\mathbf{x},\Gamma)$ is the corresponding backward and forward difference operators

$$D_{i,j}^{-x} d(\mathbf{x},\Gamma) = \frac{d(X_{i,j}) - d(X_{i,j-1})}{h_x}, \qquad D_{i,j}^{+x} d(\mathbf{x},\Gamma) = \frac{d(X_{i,j+1}) - d(X_{i,j})}{h_x}.$$

We see that (4.9) automatically chooses the direction where the derivative increases, and hence, ensures that information flows from small to big distances, meaning from the curve and outwards.

The overall strategy is to mark all grid points with known distance values in a list K, calculate the distances in their neighboring points using the upwind scheme (4.9), and store them in a data structure, T. The list K stands for *known* and T for *trial*. The list of trial points must contain both the grid point and trial value.

Since all information travels from lower to higher values, the smallest value of T must be independent of all the other points in T, and it must have the correct value. Hence, the strategy is to continually move the lowest point in T to K and recalculate its neighbors. We summarize this in a pseudo Algorithm 6, below.

Algorithm 6: The Fast Marching Method for Distance Functions

$d(X;\Gamma) = \texttt{zeros}(N,M)$;
$T = \texttt{Array\{ (i, j) : 0\}}$ for $(i,j) \in \Gamma$;
$K = \texttt{Array \{\}}$
while $T \neq \{\}$ **do**
  $(i,j) = \texttt{argmin}(T)$;
  $u_d(i,j) = T(i,j)$;
  $K \mathrel{+}= (i,j)$;
  $T \mathrel{-}= (i,j)$;
  $N = \texttt{neighbors}(i,j) \notin K$;
  for $(p,q) \in N$ **do**
    $T \mathrel{+}= (p,q) : (\texttt{solve (4.9)})$
  end
end
Result: $d(\mathbf{x};\Gamma)$

For a rectangular grid, each grid point can at most be recalculated four times because it only has four neighbors.Consequently it will have a worst-case complexity bounded by $\mathcal{O}(4(N_x \times N_y))$, which can be significantly smaller than the alternative $\mathcal{O}((N_x \times N_y) \times (N_x + N_y))$. Consequently, the fast marching method becomes increasingly superior for large point sets or equivalently increasing grid resolution when calculating the distance function to curves.

4.6 Stopping Criterion

Moving on to the final step in the third phase, we discuss a possible stopping criterion for the program. We are looking for a stationary curve, making it natural to check if the curve is still moving. However, the curve is only implicitly defined by the higher dimensional function, and one would have to find the zero contour explicitly to compare curves directly.

As stated in Section 4.5, there are packages in Python that lets you extract an approximated zero level curve, but we will see now that after re-initialization, a more convenient strategy exists. After re-initialization, the curve is a pure signed distance function to the zero level set. Since the zero-contour uniquely defines the signed distance function, we could now compare higher dimensional functions instead of zero level curves.

We denote the re-initialized function U^n. We have that $n = ls$, s is the number of time steps before re-initialization, and l denotes the lth time we re-initialize. A stopping criterion measuring how much the curve has moved since last re-initialization could be defined as follows

$$\|U^{ls} - U^{(l-1)s}\|_2 < \varepsilon, \tag{4.10}$$

for some tolerance $\varepsilon > 0$. Note that U is a matrix but in the calculations the matrices are flattened to vectors before taking the two-norm. If (4.10) is fulfilled, we announce the curve as stationary and stop the program. If not, we proceed to phase two for further time integration.

Even though this is a working stopping criterion that would detect stationary curves, we will see in the results that almost all presented examples end in oscillating curves. This is because even though the curves seem stationary macroscopically, they keep on having non-zero speed, and the stopping criterion is never triggered. A time limit is implemented to end the simulations, which in terms of a stopping criterion checks if $t^n > t_{bound}$. Since this does not require any re-initialization, it is implemented inside the inner loop before every time integration step.

Chapter 5

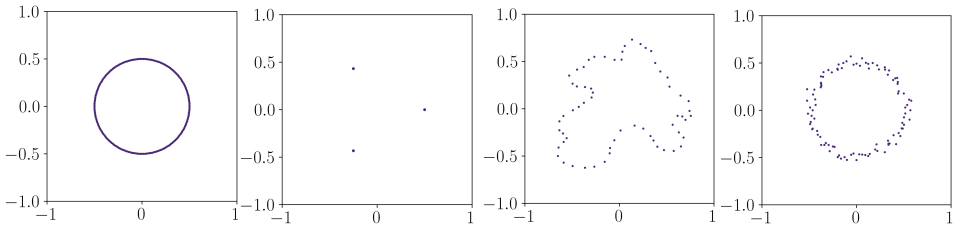
Numerical Results and Discussion

We present numerical results based on the implementation suggested in Chapter 4 for the models derived in Chapter 3. We construct four sets of test cases to demonstrate the strengths and weaknesses of the proposed models. All three models will be tested on all test cases, and we present the resulting curves and information about the transitional movement from the initial to the final solution.

Since the objective of this thesis is primarily to investigate level set methods, we have no real-world data and no objective quality measures for our solution. It is consequently not possible to compare the obtained results with an optimal solution. The presented curves will thus only display the effect of the model parameters and show how the models treat the test cases. This chapter aims to show how to adjust the presented models to obtain shapes with different features. Understanding the models will also be useful if one were to construct new velocity functions based on the presented theory.

5.1 Presentation of Test Cases

Each separate test case intends to show the models' specific traits, but the overall goal is to illustrate how they perform for irregular and noisy data sets. Furthermore, since we have performed some theoretical analysis, the



(a) Circle **(b)** Three points **(c)** Irregular shape **(d)** Noisy circle

Figure 5.1: The four sets of sample points used to produce the results presented in this chapter.

test cases should also test the compliance between the theory and the numerical simulations. Consequently, we have constructed four test cases that separately display different qualities, but together demonstrate how well the models fulfill the thesis's objective. Finally, the data sets for all test cases are presented in Figure 5.1, and we will go through the reasoning behind each test.

The first point set is displayed in Figure 5.1a, where 200 data points are distributed in a circle of radius $r_v = 0.5$. This test case is constructed to approximate the setting for the radially symmetric analysis in Chapter 3 and ergo to examine the numerical simulations compared to the theoretical results.

The point set displayed in Figure 5.1b consists only of three points with equal spacing. These points are also distributed in a circle of radius $r_v = 0.55$. The purpose of this example is to show how the solution will look if the spacing between the sample points is relatively large. Larger spacing will also show what kinds of shapes we obtain if there are few measurements.

In Figure 5.1c the points are distributed dense enough, such that the underlying shape is imaginable. Thus, we can, to some degree, envision how an optimal solution should look. At the same time, the shape includes concave and convex regions of different curvatures, which is interesting to see how the models treat.

The last test case is presented in Figure 5.1d and demonstrates how the models handle noise in the data. The radius is $r_v = 0.5$, identical to the first test case, but noise is added by sampling a normal distribution with a standard deviation of $SD = 0.08$.

	Model 1	**Model 2**	**Model 3**
k	10^{-3}	$5 \cdot 10^{-5}$	$5 \cdot 10^{-5}$
s	50	50	50

Table 5.1: k: size of time steps, s: number of time steps between re-initializations.

	Test Case 1	**Test Case 2**	**Test Case 3**	**Test Case 4**
$h_x = h_y$	0.01	0.01	0.01	0.01
R	200	3	62	100
r_v	0.5	0.55	–	0.5
r_0	0.9	0.9	0.9	0.9

Table 5.2: h_x, h_y: grid size, R: number of data points, r_v: radius of the circle which the point sets are sampled from (center in the origin), r_0: radius of the initial curve (center in the origin).

All parameters apart from the model parameters α, β and δ is not going to be tested, and the values of the parameters are found in 5.1 and Table 5.2.

5.2 Test Case 1: Circle of Dense Points

This example aims to test the numerical implementation against the analytical results for all test cases. We will, in addition, begin to test the different model parameters. Seven tests will be performed in total; two simulations for models 1 and 3, and three simulations for model 2.

For every simulation, we present a *displacement plot* and a plot of the radius of the curve during the curve evolution. The displacement plot presents a measure of the curve movement since the last re-initialization, and an example can be viewed in Figure 5.2 to the right. It is calculated as in (4.10) by flattening the matrices containing the higher dimensional functions and calculating the two-norm of the difference. Thus, it provides information about how close the curve is to a stationary solution at all times. Concretely, the value of the displacement plot will vary with the mean velocity over the last $s = 50$ time steps.

Moreover, a contour plot will be presented for all models. The contours are zero contours at different times, and the time is represented as the opacity

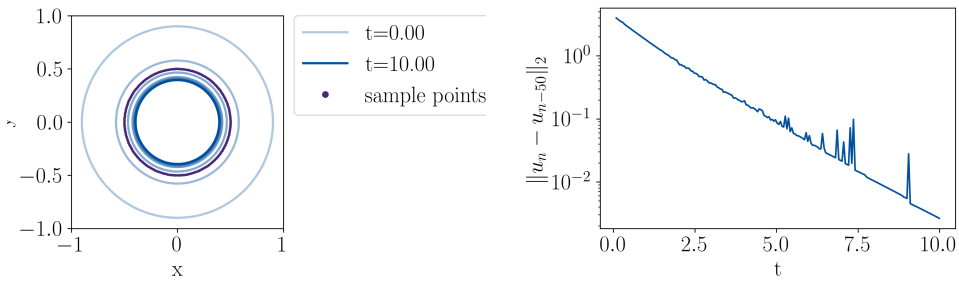


Figure 5.2: Curve evolution and displacement plot for **model 1** on the circular test case displayed in Figure 5.1a. The contour plot shows 10 level curves and the parameters for this simulation are as stated in Table 5.1 and Table 5.2 and $\alpha = 0.96$.

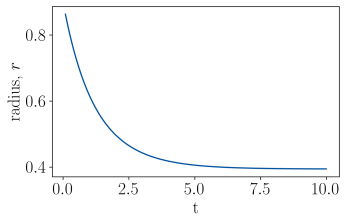
of the plot. The first contour is the lightest and is always the initial curve. The time of the first and last contour are displayed in the legend to the right. The curves are sampled uniformly in time and all plots contain 10 level curves. Hence the spacing and color of the curves will also provide an impression of the velocity during the evolution.

5.2.1 Model 1

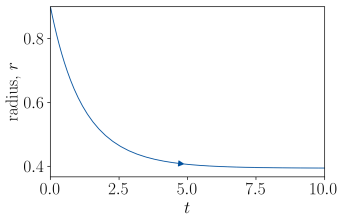
We begin with model 1, where we want to test the simulations against the analytical results and how the resulting curves depend on α. We thus run two simulations with $\alpha = 0.96$ and $\alpha = 0.99$. We expect from the theoretical analysis and the streamline plot in Figure 3.4, that the stationary radius will be closer to $r_v = 0.5$ for $\alpha = 0.99$ than for $\alpha = 0.96$. Actually, we can use equation (3.24) to find the expected radius of the stationary curve and get for $\alpha = 0.96$: $r_f = 0.39433$ and for $\alpha = 0.99$: $r_f = 0.4789$.

The results of the simulations are found in Figure 5.2 and Figure 5.3. We see by comparing Figure 5.3a and Figure 5.3b that the evolution of the curve follows the theoretical evolution perfectly. We also calculate the radius of the last zero level curve and find $r_{96}(10) = 0.3941$, which is not far from the theoretical stationary radius. We see an error in the fourth decimal place, but this is not discouraging as the discretizations of the differential operators are only second order.

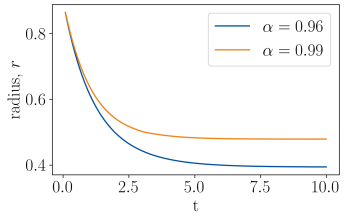
Furthermore, we see in Figure 5.3c that when $\alpha = 0.99$, the curve converges to a radius closer to the point set, as expected. The radius of the last zero level curve was calculated to be $r_{99}(10) = 0.4788$, which is very



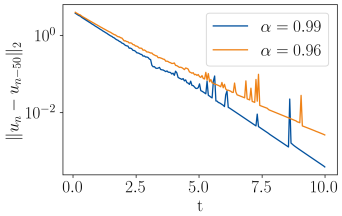
(a) Numerical solution, $\alpha = 0.96$



(b) Analytical streamline, $\alpha = 0.96$



(c) Numerical solution, $\alpha = 0.99$



(d) Displacement plot

Figure 5.3: Results for **model 1** on the test case displayed in Figure 5.1a for $\alpha = 0.96$ and $\alpha = 0.99$ and with remaining parameters as stated in Table 5.1 and Table 5.2.

close to the theoretical value.

The contour plot to the left in Figure 5.2 stems from the simulation of $\alpha = 0.96$. We see that the level curves move symmetrically towards the point set, and the speed slows down gradually during the evolution. The same is seen to the right in Figure 5.2, where the displacement plot implies a velocity going to zero, meaning a curve approaching a stationary solution. The displacement plots for both simulations can be found in Figure 5.3d where we see that increasing α means slowing down the curve. This is natural as the curvature is the main drive close to the point set.

5.2.2 Model 2

For the second model, we present the results from three different simulations. We want to test the behavior against the theoretical results and test the effect of the bounding parameter δ. In addition, we want to see the differences between models 1 and 2.

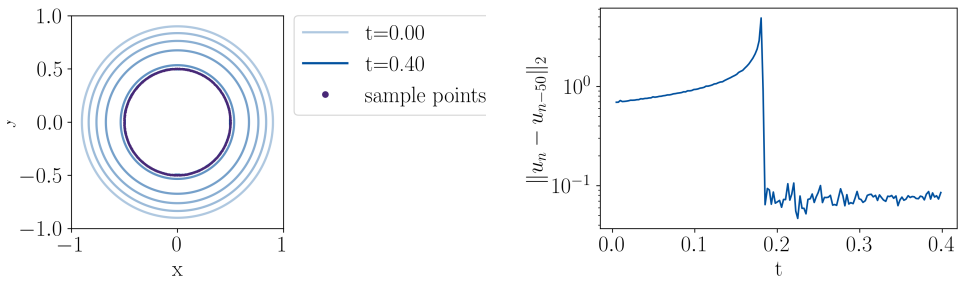
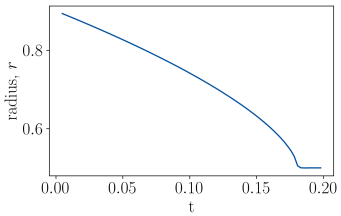


Figure 5.4: Curve evolution for **model 2** on the circular test case displayed in Figure 5.1a. The contour plot shows 10 level curves and the parameters for this simulation are as stated in Table 5.1 and Table 5.2 and $\alpha = 0.2$, $\beta = 1$, $\delta = 10^{-2}$.

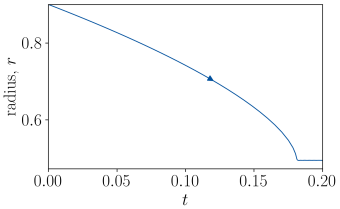
We expect that the zero level curve for model 2 will approach the point set, moving faster and faster on the way in. From the streamlines in Figure 3.6 we saw that the velocity had a discontinuity at the radius of the point set. The velocity would be big in the inward direction when $r_\Gamma > r_v$ and big outwards when $r_\Gamma < r_v$. Since we saw that the curve would never get a stationary solution when $r_\Gamma = r_v$, we expect oscillations for the numerical simulations. The amplitude of the oscillations will be decided from the time step size, k, and the bounding parameter δ. The parameter δ decides the velocity, and k decides how far the curve moves between every time step. We want to check this hypothesis by varying the parameter δ and observing the oscillations' size.

We see the evolution of a simulation using $\alpha = 0.2$, $\beta = 1$, $\delta = 10^{-2}$ in Figure 5.5a and compare with the analytical evolution using the same parameters in Figure 5.5b. The motion is almost identical up until the curve hits the point set. We see that the numerical simulation seems slower at exactly that moment. However, the radius is only calculated every 50th time steps, and hence poor resolution is the most probable cause of the smoothened line. In the contour plot to the left in Figure 5.4 we also see that the curve speeds up near the point set, as the spacing between contours is larger here than in the beginning. The evolution is also completely symmetric, which is as expected for the symmetric problem.

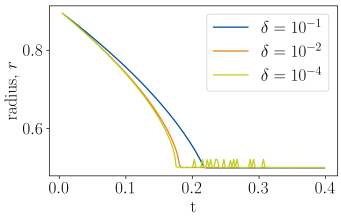
Now, we discuss the result of the comparison of different values of δ found in Figure 5.5c and Figure 5.5d. We see in the plot of the radii that the motion is very similar for $\delta = 10^{-2}$ and $\delta = 10^{-4}$. The bound δ is very small compared to the distance until $r \sim r_v$, and it is thus expected that the differences are small. For $\delta = 0.1$, the bound is more influential earlier in the evolution, and the curve moves slower than for the other tests. The



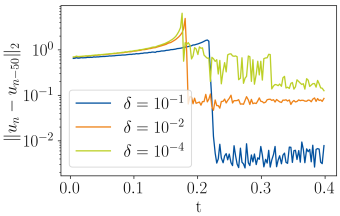
(a) Numerical solution, $\alpha = 0.2$, $\beta = 1$, $\delta = 10^{-2}$



(b) Analytical streamline, $\alpha = 0.2$, $\beta = 1$, $\delta = 10^{-2}$



(c) Numerical solution, varying δ



(d) Displacement plot, varying δ

Figure 5.5: Results for **model 2** on the test case displayed in Figure 5.1a for $\delta = 10^{-1}$, $\delta = 10^{-2}$ and $\delta = 10^{-2}$. $\alpha = 0.2$, $\beta = 1$ for all simulations and the remaining parameters are as stated in Table 5.1 and Table 5.2.

speed at the curve is also reflected in the displacement plot where we, as expected, see that the curve has bigger oscillations for smaller δ.

We end with a discussion concerning the displacement plot in Figure 5.5d compared to model 1 in Figure 5.3d and what they mean for a stopping criterion. For model 1, we saw that the displacement kept shrinking near the point set, and a stopping criterion could easily be implemented almost no matter the choice of ε. For model 2, on the other hand, it could be very challenging to choose the right tolerance. Look at Figure 5.5d for $\delta = 10^{-4}$ for instance. Choosing $\varepsilon \gtrsim 0.8$ and the stopping criterion is triggered instantly and choosing $\varepsilon \lesssim 0.1$ and the stopping criterion is never triggered at all.

5.2.3 Model 3

Moving on to model 3, the model parameter left to test is β. Similar to the other models, we run a comparison between the numerical simulations and the analysis performed in Chapter 3. In addition, it is interesting to compare models 2 and 3 since their velocity functions are similar. Hence, we have run two simulations with $\beta = 2$ and $\beta = 8$ to see the parameter's effect properly.

As for model 2, we expect the curve to move faster and faster inwards to the point set, where it should suddenly stop. Since the velocity function is discontinuous in time here, we believe that there will be oscillations around the radius of the point set. A relevant difference between the models 2 and 3 is how they are bounded when $d(\mathbf{x}; \mathcal{V}) \to 0$. We saw that the bounding parameter, δ, was very important for the behavior of model 2 in Section 5.2.2. For model 3, the distance term f_3 in (4.3) is naturally bounded since $f_3 \to 1$ as $d \to 0$ and thus we do not expect β to influence the amount of oscillations. Furthermore, this means that the maximum velocity should be equal for the two test runs. However, how the speed increase is expected to differ since parameter β scales the distance.

We see the results of the simulations in Figure 5.6 and Figure 5.7. First, we observe from the radius plot from the numerical simulations, displayed in Figure 5.7a compared to the analytical streamline in Figure 5.7b, that the curve evolution coincides with the analytical streamline. The curves also move symmetrically inwards in the contour plot to the left in Figure 5.6 as expected. We see that the curves are moving with what looks like constant

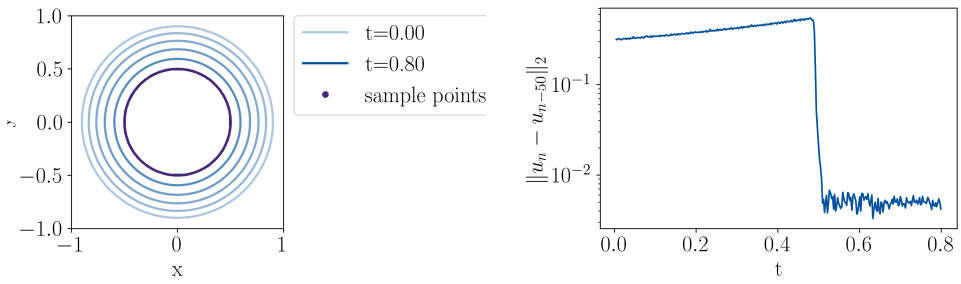
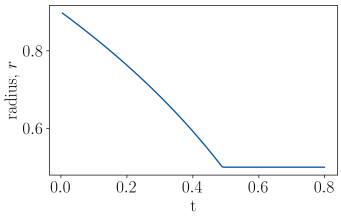


Figure 5.6: Curve evolution for **model 3** on the circular test case displayed in Figure 5.1a. The contour plot shows 10 level curves and the parameters for this simulation are as stated in Table 5.1 and Table 5.2 and $\alpha = 0.9$, $\beta = 2$.

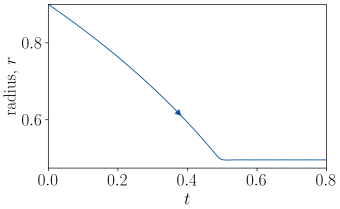
speed because the spacing of the contours is almost equal. This can also be seen by how the radius changes in Figure 5.7a.

We move on to the effect of the parameter β. As predicted, it has no clear effect on the oscillations after the curve has reached the point set. In addition, observe that the maximum velocity is unaffected, as expected. An important observation is that when the distance is scaled up, the velocity far away is smaller, but the acceleration is bigger near the point set. This comes directly from the inverse relation to the distance. As a result, we see that if we start sufficiently far from the sample points, or β is big enough; the velocity would be minimal. The small velocity would complicate a stopping criterion.

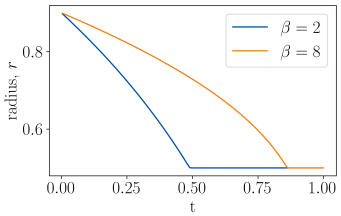
Fortunately, the velocity would not become zero, and the curvature and distance would draw in the same inward direction. Hence, the curve would reach the point set, but the first phase of the evolution would be very slow. This observation also holds for model 2.



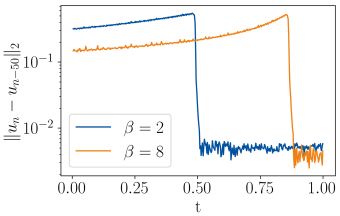
(a) Numerical solution, $\alpha = 0.9$, $\beta = 2$



(b) Analytical streamline, $\alpha = 0.9$, $\beta = 2$



(c) Numerical solution, varying β



(d) Displacement plot, varying β

Figure 5.7: Results for **model 3** on the test case displayed in Figure 5.1a for $\beta = 2$ and $\beta = 8$. $\alpha = 0.9$ for all simulations and the remaining parameters are as stated in Table 5.1 and Table 5.2.

5.3 Test Case 2: Three Equidistant Points

We saw from test case 1 that the simulations were consistent with the theoretical analysis and expectations. Unfortunately, we do not have any theoretical results for this test case to compare with, so we rely only on the numerical simulations alone.

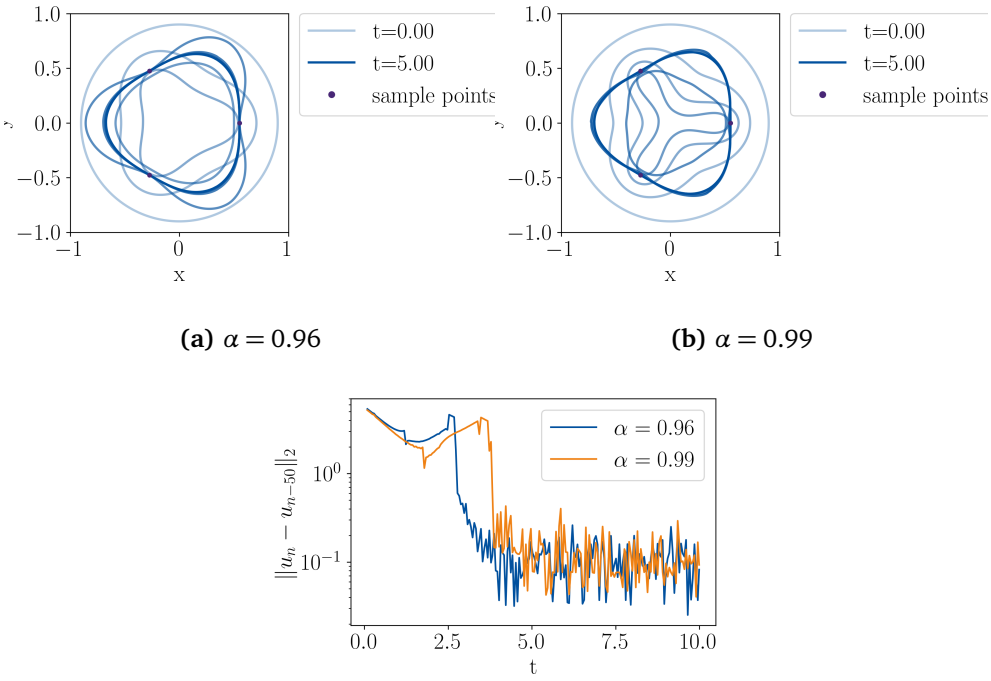
We will investigate further the differences between the models and how adjusting the parameters changes the evolution and the final curves. We will, in addition, see an example of what can happen if the parameters are not chosen carefully, and there is no final curve. In the first test case, the problem was completely symmetric, and the point set was very dense. This gave little flexibility for different curve shapes. We will now see much more interesting contour plots, and contours are presented for all simulations. Now that there are only three points, the curves are not expected to be symmetric around the origin, and no plots of the radius are presented. There are carried out six simulations in total, two for each model.

5.3.1 Model 1

We will run two simulations with $\alpha = 0.96$ and $\alpha = 0.99$. The aim is to see how α can affect the curve shape and how the curvature differs over the curve when there are few sample points.

As observed when we introduced model 1 in Section 3.1 the stationary solution would not minimize both the curvature and the distance, but it will yield a curve balancing the two. When the distance is low, the curvature will be low and vice versa. Hence, we expect to see curves with low curvature near the points and increasing curvature as the distance to the point increases. When α is changed, the balance is shifted. We expect increased curvature further away from the data points for $\alpha = 0.99$ than for $\alpha = 0.96$.

We see the resulting curve evolution in Figure 5.8 for both simulations. The final curves have the darkest color and fulfill the expectations, having low curvature close to the points and higher curvature further away. It is, however, challenging to see clear differences between the final curves for the two simulations. This is most likely due to the oscillating behavior we see in Figure 5.8c, which comes when the requirement for zero velocity is not fulfilled. We predicted differences in the stationary solutions, but we



(a) $\alpha = 0.96$

(b) $\alpha = 0.99$

(c) Displacement plot for $\alpha = 0.96$ and $\alpha = 0.99$

Figure 5.8: Results for **model 1** on the test case in Figure 5.1b for $\alpha = 0.96$ and $\alpha = 0.99$ and the remaining parameters as stated in Table 5.1 and Table 5.2.

never actually obtain the stationary curves.

The biggest differences are seen in the evolution of the curves. When α is big, the curve sections furthest away move fastest, and the result is seen in Figure 5.8a and Figure 5.8b. In Figure 5.8b the curves almost meet before the curve crosses the sample points, and the sign function draws the curve outward again. For an even increased α, the curves will meet and split the curve into smaller closed sub-curves. This would not yield a meaningful final solution.

Observe further that multiple zero contours are touching the three sample points. They have different shapes, but since we have no prior information about the curve shape, any of these solutions would be acceptable. When we now move on to model 2, we will see that the curves will be more similar to the innermost curve in Figure 5.8a. This contour is the most similar to the minimal curve going through the sample points.

5.3.2 Model 2

There will be run two simulations for model 2. The purpose is to see how easily a curve can be transformed from what seems like a stationary curve to a meaningless solution. More concrete, the parameter α will be tested, and we will see what happens if it is chosen too small and the curvature dominates. The tests are for $\alpha = 0.4$ and $\alpha = 0.2$.

When α decreases, the balance is shifted, and the curve flows towards low curvature. In practice, this means that curves with high curvatures will have higher velocity inwards. However, model 2 is constructed to have high curvature close to the points, and it will thus be vulnerable for too low α near the data points.

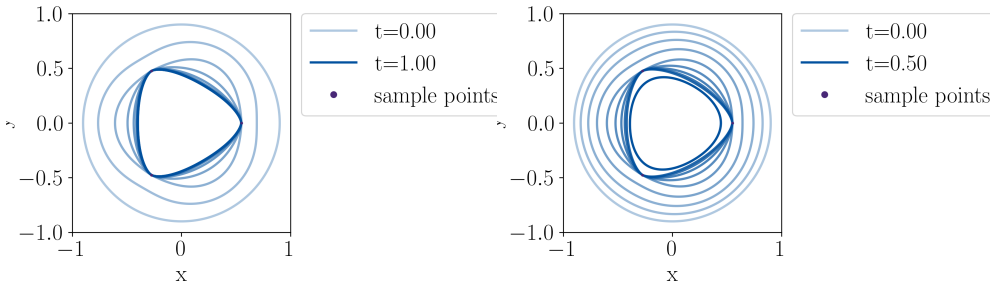
We see the results of the simulations in Figure 5.9. For $\alpha = 0.4$, we see a nice final solution going through the sample points while resembling a minimal curve. We can see from the contours in Figure 5.9a that the five last contours are inseparable, and the cure is in practice stationary. There are in total ten curves in both figures. By counting the curves in Figure 5.9b, we see that there are two inseparable curves also for $\alpha = 0.2$ that resemble the solution for $\alpha = 0.4$. However, the curvature is too strong for this example, and the curve slips the points and falls inwards. Note that the curve is stopped at $t = 0.5$ because the curve moves very fast towards the middle and disappears. Without the curve, re-initialization is not possible, and the simulation breaks.

We see when we compare the displacement plots in Figure 5.9c and Figure 5.9d that the evolution of the curves are very similar. The curve in Figure 5.9b is very close to obtaining its final solution at the point set, and there is no way to see that the simulation will go wrong before the curve slips.

We have thus seen that the models are very sensitive for alpha. Further, note that since the distance grows large close to the sample points, α needs to be smaller for model 2 than model 1.

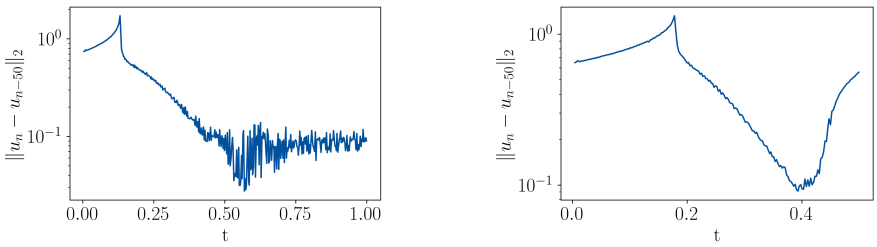
5.3.3 Model 3

For model 3, we will see that the parameter β is very important to construct different shapes. Two simulations with different choices of β are



(a) Curves, $\alpha = 0.4$

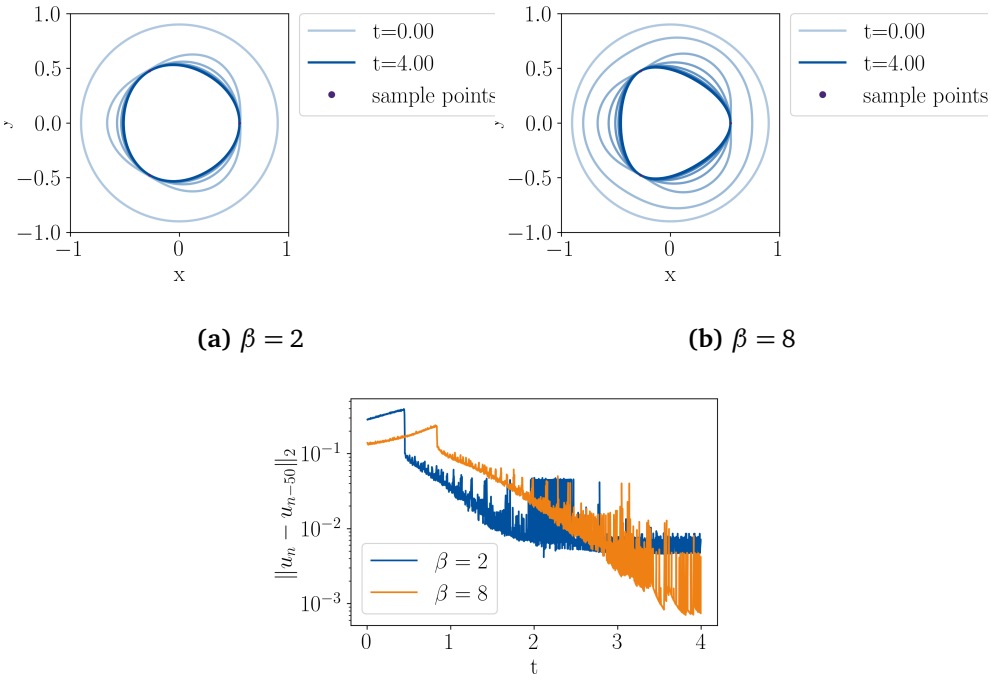
(b) Curves, $\alpha = 0.2$



(c) Displacement plot, $\alpha = 0.4$

(d) Displacement plot, $\alpha = 0.2$

Figure 5.9: Results for **model 2** on the test case displayed in Figure 5.1b for $\alpha = 0.4$ and $\alpha = 0.2$ and fixed model parameters $\beta = 1$ and $\delta = 10^{-2}$. The remaining parameters are as stated in Table 5.1 and Table 5.2.



(a) $\beta = 2$ **(b)** $\beta = 8$

(c) Displacement plot for $\beta = 2$ and $\beta = 8$

Figure 5.10: Results for **model 3** on the test case in Figure 5.1b for $\beta = 2$ and $\beta = 8$. $\alpha = 0.9$ for both simulations and the remaining parameters as stated in Table 5.1 and Table 5.2.

constructed to show this. Furthermore, the simulations show how similar model 3 can look to model 2 while still having differences.

The parameter β scales the distance function for model 3. If $\beta = 2$ that is equal to doubling the domain, $\mathcal{D}$, and doubling the spacing between the points. The relation between the curvature and distance is that the further away from the points, the smaller the curvature. It follows that scaling up β will lead to lower curvature between the points. If β is set extremely high, the curvature decreases to zero almost instantly, leading to a curve similar to a polygon.

We see the results of the two simulations in Figure 5.10. Again, the curves move as expected. Moreover, we see that the curve for $\beta = 8$ looks much more similar to the polygon and thus model 2. It is, however, rounder in the corners.

From the displacement plot in Figure 5.10c, we see that even though model

3 with $\beta = 8$ looks similar to the simulations of model 2, the velocity is slower, which causes the oscillations to diminish. Moreover, the velocity is overall smaller when β is big because the velocity for models 2 and 3 is slower the further away.

5.4 Test Case 3: Dense and Irregular Data

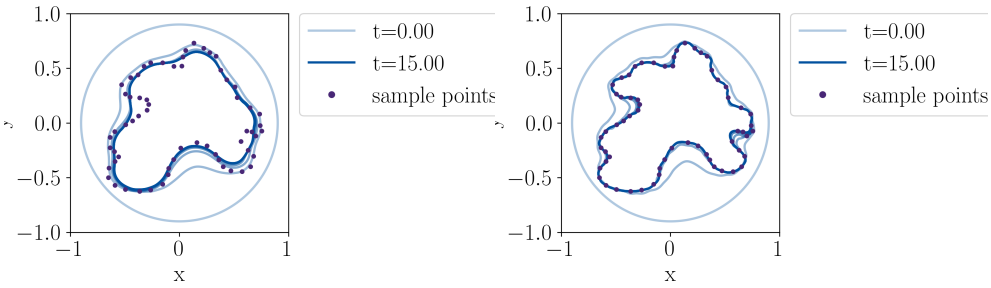
In this test case, the points are dense compared to the last example. Moreover, there is much more flexibility to form different shapes than for the first circular example. The shape also includes regions with different curvatures, and we will see that the curves will fit the data points differently from model to model. For the circular test case, the circle having a radius of $r = 0.5$ was obviously the correct answer. Test case 2 showed the complete opposite, where there were too few points to say anything about a correct reconstruction.

This test case will be a little bit of both; the data set provides some flexibility while we can still visualize a good solution. This point set can be similar to a real-world example because we know roughly what the solution should look like, but the exact curve is unknown. Therefore, we will try to find suitable parameter choices for all models to obtain nice solutions, like we would have done for real-world data. We try to find one set of parameters yielding a curve with lower curvature and one where we try to fit the data points exactly. There will, in total, be presented seven test cases; two for models 1 and 2 and three for model 3.

5.4.1 Model 1

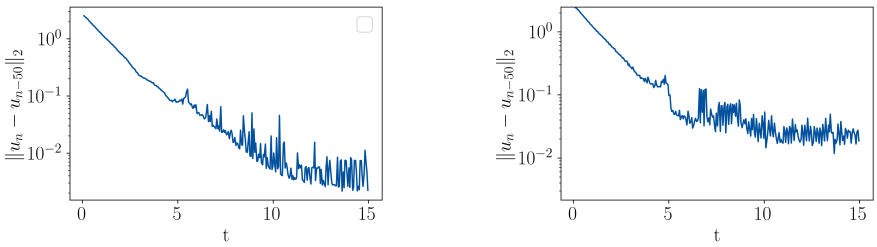
Two tests are performed with two different choices of α. This will show how solutions depending on the balance between curvature and distance attraction. The first test will have $\alpha = 0.99$, and we compare the solution with a test run with $\alpha = 0.999$. Note that both tests have a relatively high α compared to the already presented results. The distance function needs to be dominant to catch the details of the shape.

The results are shown in Figure 5.11. For the first simulation with $\alpha = 0.99$, we see that the curve quickly obtains a shape similar to the point set. The final curve fulfills the goal of approximating the sample points while having



(a) Curves, $\alpha = 0.99$

(b) Curves, $\alpha = 0.999$



(c) Displacement plot, $\alpha = 0.99$

(d) Displacement plot, $\alpha = 0.999$

Figure 5.11: Results for **model 1** on the test case displayed in Figure 5.1c for $\alpha = 0.99$ and $\alpha = 0.999$ and fixed model parameters as stated in Table 5.1 and Table 5.2.

low curvature, although the curve is slightly biased towards the inside. The bias is similar to what we saw in the circular example. A nice detail observed from this simulation is that the curve is almost a straight line up in the right corner while it seems to approximate the points well. Hence, we can assume that model 1 will be very well suited to filter noise from approximately straight lines.

For $\alpha = 0.999$ we catch much more details in the shape as we hoped, and the curves provide a nice approximation to the underlying shape. However, there is one small region where the curve does not cover the data points, which is in the middle right part of Figure 5.11b. However, this is also the curve section with the highest curvature and the most densely spaced points. Consequently, the curvature will be most prominent in this region compared to the distance function.

The residual plots for $\alpha = 0.99$ and $\alpha = 0.999$ in Figure 5.11c and Figure 5.11d shows that both curves have a small velocity at the end of the simulations. Furthermore, smaller α leads to a more curvature-driven flow and an even more decreasing velocity. The curvature always pulls in the same direction, and the curve will not be drawn back and forth over the points to the same extent. It is the sign-changes that cause the oscillations. Thus, a less distance-dependent velocity leads to smaller oscillations.

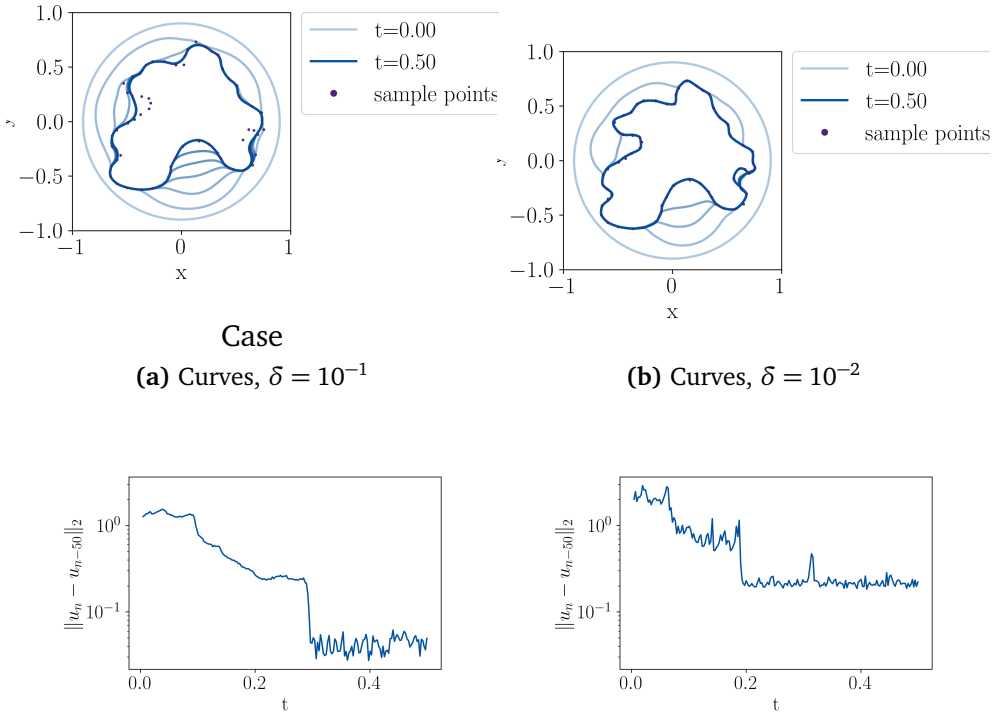
5.4.2 Model 2

For model 2, we perform two tests, which collectively show that changing the parameter δ can vary the degree of smoothing. Note that smoothing was performed solely by α for model 1 discussed above. The first test has $\delta = 10^{-1}$, and for the second case, we decrease it to $\delta = 10^{-2}$.

In order to predict the δ-dependency for model 2, we look at the model equation restated in (4.2) with $u_t = 0$, meaning stationary solution. We have

$$(1-\alpha)\kappa(u) = \pm\frac{\alpha}{\beta d(\mathbf{x};\mathcal{V}) + \delta},$$

where the sign is decided by the sign function. We see that by decreasing δ, the right hand side increases and the curvature increase. From this short argument, we expect a final curve with higher curvature for $\delta = 10^{-2}$ than for $\delta = 10^{-1}$. This is also what we see in the contour plots in Figure 5.12a and Figure 5.12b. The displacement plots in Figure 5.12c and Figure 5.12d further show that we still get higher velocity oscillations for smaller δ.



(a) Curves, $\delta = 10^{-1}$

(b) Curves, $\delta = 10^{-2}$

(c) Displacement plot, $\delta = 10^{-1}$

(d) Displacement plot, $\delta = 10^{-2}$

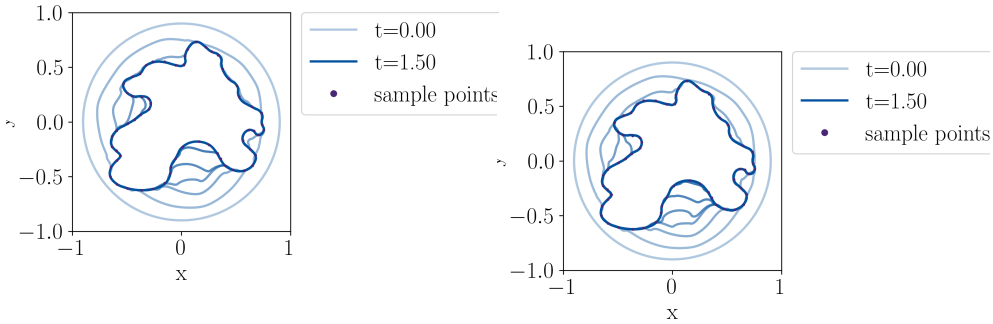
Figure 5.12: Results for **model 2** on the test case displayed in Figure 5.1c for $\beta = 1$ and $\beta = 0.7$ and fixed model parameters $\delta = 10^{-1}$ and $\delta = 10^{-2}$ and the remaining parameters $\alpha = 0.5$, $\beta = 1.2$ and the constant parameters as stated in Table 5.1 and Table 5.2.

5.4.3 Model 3

There will be run three simulations for model 3 for this test case, and the simulations aim to present good parameter choices. We vary both model parameters α and β and observe the effects on the resulting curves.

We expect to see that α affects the curvature similarly to what we saw for model 1 on the same example. The question is how β will influence the solution. Remember that β is the scaling of the distance function for the model. Since the curve moves faster when closer to the point set, a down-scaling would make the distances smaller, and the curves would move faster.

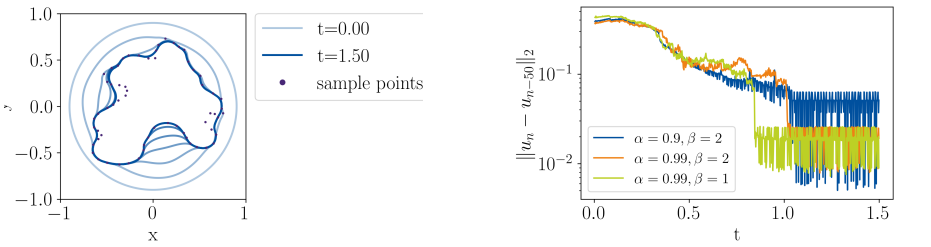
The effects are seen in Figure 5.13. A smaller α smooths the solution in Figure 5.13c and increasing α in Figure 5.13b compared with Figure 5.13a leads to a slower evolution. The speed difference can be seen both in the contour plots from the spacing of the curves and more clearly in the displacement plot in Figure 5.13d.



Case

(a) Curves, $\alpha = 0.99$, $\beta = 1$

(b) Curves, $\alpha = 0.99$, $\beta = 2$



(c) Curves, $\alpha = 0.9$, $\beta = 2$

(d) Displacement plot for all simulations

Figure 5.13: Results for **model 3** on the test case displayed in Figure 5.1c for $alpha = 0.9$ and $\alpha = 0.99$, and for $\beta = 1$ and $\beta = 2$. The remaining constant parameters are as stated in Table 5.1 and Table 5.2.

5.5 Test Case 4: Noisy Data

The noisy data set and the dense and irregular data set in test case 3 are together constructed to give an impression of how well the models achieve the goals set for this thesis: to reconstruct curves from irregular and noisy data. The data points are distributed in a circle with a radius of $r = 0.5$, just like in test case 1. However, there is added visible noise. The example is designed this way to be able to compare the models from tests 1 and 4, while the noise is visible enough to see clear differences. There are performed two simulations for each model, which makes six simulations in total.

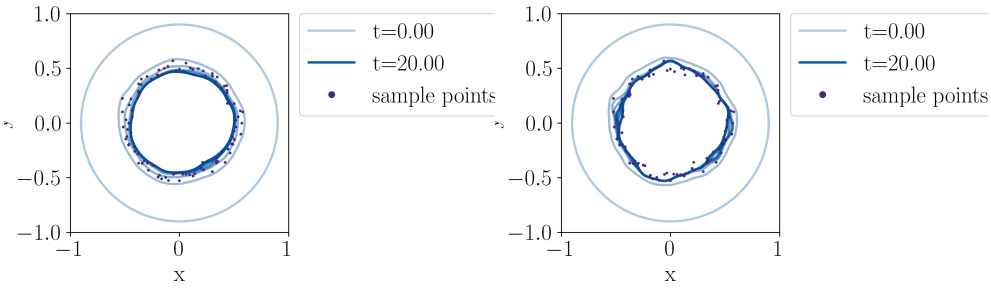
5.5.1 Model 1

We have seen in the earlier tests that the parameter α can be adjusted to increase or decrease the curvature for model 1. For the first circular example, we also saw that decreasing α would lead to a curve falling inside the point set. Specifically for $\alpha = 0.96$ in test case 1, the end radius was distinctly inside the point set. For this reason, both simulations are run with relatively high values of α to obtain better approximations.

The tests are run with $\alpha = 0.99$ and $\alpha = 0.995$. Since $\alpha = 0.99$ had an analytical stationary radius of $r_f = 0.479 < 0.5$ with no noise, we expect the curve to be slightly biased inward. We further calculate the analytical radius without noise for $\alpha = 0.995$, and get $r_f = 0.491$. This is still inside the point set, but we expect a smaller bias for this simulation. However, we expect a smoother curve with lower curvature for $\alpha = 0.99$ than $\alpha = 0.995$, so the choice of α is a trade-off between low curvature and a good approximation, as we have also seen in the earlier results.

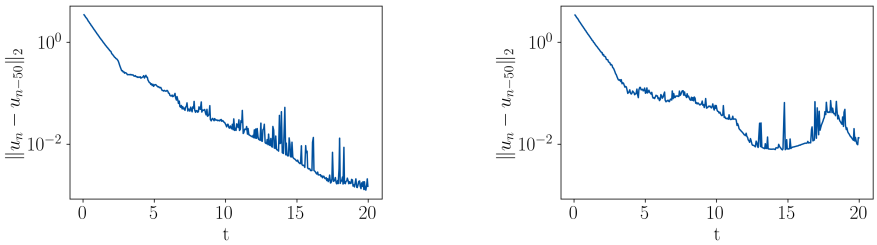
We see the results in Figure 5.14a and Figure 5.14b, and we observe the expected difference in curvature. We have also calculated the end radius for the curves and got $r_{99}(20) = 0.4643$ and $r_{995}(20) = 0.5011$. The resulting curves display the importance of α and how sensitive the model is to a small change in the parameter.

For $\alpha = 0.996$ in Figure 5.14b, we observe that the final curve catches some of the patterns in the random noise. At the bottom, the points are located densely and slightly further from the center, and the final curve covers these points. We can also see from the final radius that this simulation is



(a) Curves, $\alpha = 0.99$

(b) Curves, $\alpha = 0.996$



(c) Displacement plot, $\alpha = 0.9$

(d) Displacement plot, $\alpha = 0.999$

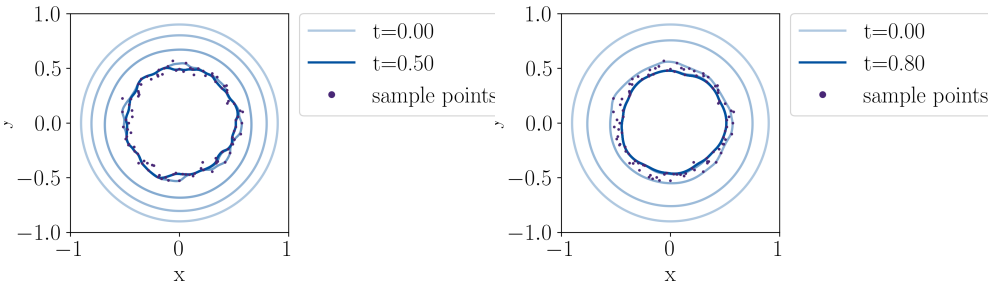
Figure 5.14: Results for **model 1** on the test case displayed in Figure 5.1d for $\alpha = 0.99$ and $\alpha = 0.995$ and fixed model parameters as stated in Table 5.1 and Table 5.2.

not biased inwards as we expected. That is explained by the local behavior of the level set models; the curve is only attracted to its closest points. Thus, when the points have noise, and the curve is on its way in, it will be attracted to the furthest points from the center. If the attraction is big enough, the curve will stop here, leading to a bias outward.

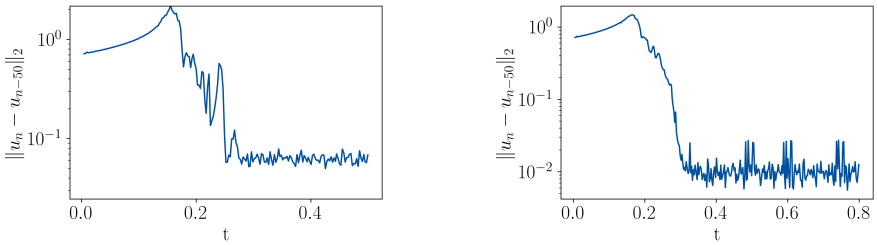
5.5.2 Model 2

We have seen that changing the parameter δ makes the curves smoother. It bounds the distance-dependent function $f_2(d(\mathbf{x}; \mathcal{V}))$ and, hence, the curvature. We will run two simulations in order to see if we can produce a smooth curve by varying δ for model 2.

In the first test run, $\delta = 10^{-2}$ and $\alpha = 0.2$. The parameter α must be low because the function $f_3(d(\mathbf{x}; \mathcal{V}))$ is allowed to grow big. The second



(a) Curves, $\alpha = 0.2$, $\delta = 10^{-2}$ **(b)** Curves, $\alpha = 0.3$, $\delta = 10^{-1}$



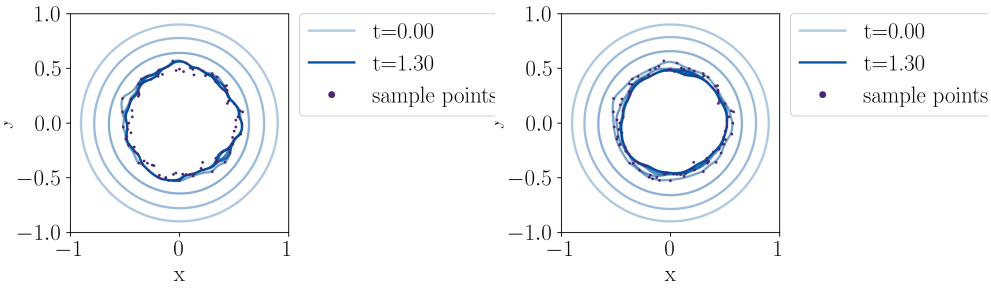
(c) Displacement plot, $\alpha = 0.2$, $\delta = 10^{-2}$ **(d)** Displacement plot, $\alpha = 0.3$, $\delta = 10^{-1}$

Figure 5.15: Results for **model 2** on the test case displayed in Figure 5.1d for $\alpha = 0.2$ and $\delta = 10^{-2}$, and $\alpha = 0.3$ and $\delta = 10^{-2}$ and fixed $\beta = 1$ and remaining model parameters as stated in Table 5.1 and Table 5.2.

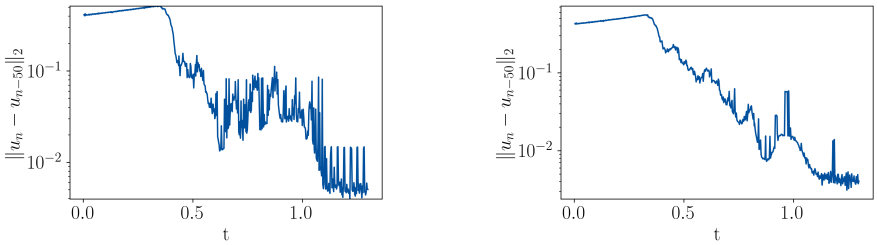
simulation is for $\delta = 10^{-1}$ which bounds f_3 and we increase α to $\alpha = 0.3$ to prevent an entirely curvature driven evolution.

If we only consider the increase in α, we would expect the smoothest curve for the first simulation. However, since the points are densely spaced, the distance-dependent function f_3 is mainly decided by the bound $1/\delta$. It is consequently a combination of α and δ that decides the curvature. Therefore, we expect a considerably smoother curve for the second test where we have decreased the bound from $f_3 < 100$ to $f_3 < 10$.

The results are shown in Figure 5.15 and we see from the contours in Figure 5.15a and Figure 5.15b that the curve is as expected noticeably smoother for higher δ. In Figure 5.15b we see similarly to model 1, that the curve is slightly biased towards the inside of the sample points. Increasing α further is the strategy to mend this, but as we know, this would also affect the curvature.



(a) Curves, $\alpha = 0.9$ **(b)** Curves, $\alpha = 0.8$



(c) Displacement plot, $\alpha = 0.9$ **(d)** Displacement plot, $\alpha = 0.8$

Figure 5.16: Results for **model 3** on the test case displayed in Figure 5.1d for $\alpha = 0.9$, and $\alpha = 0.8$, and fixed $\beta = 1$, and remaining model parameters as stated in Table 5.1 and Table 5.2.

5.5.3 Model 3

We saw in the last test case that the scaling parameter β had little effect on the curvature when the distance to the points is small. Furthermore, adjusting the curvature is important in examples where the points are noisy and densely spaced.

We will hence adjust α to obtain two curves of different curvatures just like the two previous models. The results are presented in Figure 5.16 and the contours in Figure 5.16b shows as expected, curves with lower curvature than in Figure 5.16a. This example concludes the testing of the models, and we end with a summary of what we observed from the presented simulations.

5.6 Summary of Results

We have seen that all the models behave as expected for the circle. The parameter α scales how much the curvature affects the solution for all models. When we introduce the parameter δ for model 2, we also see that this affects the curvature of the final solution near the points. If δ is increased, it bounds the growing curvature, such that the edges are less sharp. The parameter β in models 2 and 3 does not affect the maximum sharpness of the edges, but it scales the distance. A higher β means flatter regions further away, but it does not bound the curvature close to the sample points.

We have seen that choosing suitable parameters is essential for all models, and for proper values, all models yield similar curves if the data set is dense. The difference between the models was most apparent for the point set with three points, where it was most visible how the curvature varies when the distance increases.

If we now assume that we have some a priori knowledge about the underlying shape, we can pick the sample points to represent certain qualities of the curve and then choose a model that fits. For example, if we sample the curve where the curvature is lowest, model 1 is the best choice to reconstruct it afterward. On the other hand, if the points are sampled at something similar to edges on the underlying curve, model 2 with a small δ would be preferred.

Also, model 3 has the advantage over model 2 by having fewer parameters to tune, so the behavior of the model is less complicated. However, it does not have the same flexibility to tune the curvature vs. distance relation.

Finally, notice that none of the models reached clean stationary solutions for all problems. The reason is the local behavior of the distance function, which creates oscillations if the curve goes back and forth over the curve. Nevertheless, with suitable tuning parameters, the models reached a stationary solution on a macroscopic level. The oscillations are less problematic since we assume little about the underlying shape, and the oscillations do not perturb the solution much. Since this is an approximation problem without a solution to compare with, any of the final curves would be acceptable.

Chapter 6

Concluding Remarks

We introduced three level set models and presented results based on four constructed test cases. The variety of results demonstrated flexibility, and we saw that it was relatively simple to formulate new models. In addition, by choosing the proper parameters, we obtained nice solutions for all test cases. However, the obtained solutions were not stationary but oscillated around the data points due to the alternating sign function. The oscillations did not alter the overall shape, but it made a stopping criterion hard to implement and was not the elegant solution we hoped for. Finally, it is hard to conclude how these models performed compared to other shape reconstruction methods, as these have not been studied.

6.1 Further Research

One of the aspects that we wanted to investigate further was the discrepancy between the assumptions in the theory and the introduced sign function. This issue was found relatively late in the process, but a more firm anchoring in the theory would improve the overall impression of the models. Moreover, it would be interesting to implement time discretization techniques with a step size control adjusted to handle temporal discontinuities. This would potentially reduce the oscillations drastically, and in this way, make a stepping criterion more applicable. Furthermore, we observed that model 1 improved the curves more in the early stages, but models 2 and 3 moved the curve faster in the final stages of the simulations. An idea could thus be to combine different models to take advantage of their strengths.

Bibliography

[1] J.-D. Boissonnat and B. Geiger, "Three-dimensional reconstruction of complex shapes based on the Delaunay triangulation," in *Biomedical Image Processing and Biomedical Visualization*, R. S. Acharya and D. B. Goldgof, Eds., International Society for Optics and Photonics, vol. 1905, SPIE, 1993, pp. 964–975. [Online]. Available: https://doi.org/10.1117/12.148710.

[2] N. Amenta and M. Bern, "Surface reconstruction by Voronoi filtering," in, 4, vol. 22, 14th Annual ACM Symposium on Computational Geometry (Minneapolis, MN, 1998), 1999, pp. 481–504. DOI: 10.1007/PL00009475. [Online]. Available: https://doi.org/10.1007/PL00009475.

[3] J.-D. Boissonnat and F. Cazals, "Smooth surface reconstruction via natural neighbour interpolation of distance functions," in, 1-3, vol. 22, 16th ACM Symposium on Computational Geometry (Hong Kong, 2000), 2002, pp. 185–203. DOI: 10.1016/S0925-7721(01)00048-7. [Online]. Available: https://doi.org/10.1016/S0925-7721(01)00048-7.

[4] M. G. López, B. Mederos, and O. Dalmau, "Gp-mpu method for implicit surface reconstruction," in *Human-Inspired Computing and Its Applications*, A. Gelbukh, F. C. Espinoza, and S. N. Galicia-Haro, Eds., Cham: Springer International Publishing, 2014, pp. 269–280, ISBN: 978-3-319-13647-9.

[5] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations," *J. Comput. Phys.*, vol. 79, no. 1, pp. 12–49, 1988, ISSN: 0021-9991. DOI: 10.1016/0021-9991(88)90002-2. [Online]. Available: https://doi.org/10.1016/0021-9991(88)90002-2.

[6] M. Sussman, P. Smereka, and S. Osher, "A level set approach for computing solutions to incompressible two-phase flow," *Journal of Computational Physics*, vol. 114, no. 1, pp. 146–159, 1994, ISSN:

0021-9991. DOI: https://doi.org/10.1006/jcph.1994.1155. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999184711557.

[7] H.-K. Zhao, T. Chan, B. Merriman, and S. Osher, "A variational level set approach to multiphase motion," *Journal of Computational Physics*, vol. 127, no. 1, pp. 179–195, 1996, ISSN: 0021-9991. DOI: https://doi.org/10.1006/jcph.1996.0167. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999196901679.

[8] J. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, ser. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 1999, ISBN: 9780521645577. [Online]. Available: https://books.google.no/books?id=ErpOoynE4dIC.

[9] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259–268, 1992, ISSN: 0167-2789. DOI: https://doi.org/10.1016/0167-2789(92)90242-F. [Online]. Available: https://www.sciencedirect.com/science/article/pii/016727899290242F.

[10] R. Malladi, J. Sethian, and B. Vemuri, "Shape modeling with front propagation: A level set approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 2, pp. 158–175, 1995. DOI: 10.1109/34.368173.

[11] C. Li, R. Huang, Z. Ding, J. C. Gatenby, D. N. Metaxas, and J. C. Gore, "A level set method for image segmentation in the presence of intensity inhomogeneities with application to mri," *IEEE Transactions on Image Processing*, vol. 20, no. 7, pp. 2007–2016, 2011. DOI: 10.1109/TIP.2011.2146190.

[12] A. Claisse and P. Frey, "A nonlinear PDE model for reconstructing a regular surface from sampled data using a level set formulation on triangular meshes," *J. Comput. Phys.*, vol. 230, no. 12, pp. 4636–4656, 2011, ISSN: 0021-9991. DOI: 10.1016/j.jcp.2011.02.039. [Online]. Available: https://doi.org/10.1016/j.jcp.2011.02.039.

[13] P. Baxandall and H. Liebeck, *Vector calculus*, ser. Oxford Applied Mathematics and Computing Science Series. The Clarendon Press, Oxford University Press, New York, 1986, pp. x+550, ISBN: 0-19-859652-9.

[14] R. Adams and C. Essex, *Calculus: A Complete Course*. Pearson Canada, 2009, ISBN: 9780321549280. [Online]. Available: https://books.google.no/books?id=vFdKPgAACAAJ.

[15] Y. G. Chen, Y. Giga, and S. Goto, "Uniqueness and existence of viscosity solutions of generalized mean curvature flow equations," *J. Differential Geom.*, vol. 33, no. 3, pp. 749–786, 1991, ISSN: 0022-040X. [Online]. Available: http://projecteuclid.org/euclid.jdg/1214446564.

[16] L. C. Evans and J. Spruck, "Motion of level sets by mean curvature. I," *J. Differential Geom.*, vol. 33, no. 3, pp. 635–681, 1991, ISSN: 0022-040X. [Online]. Available: http://projecteuclid.org/euclid.jdg/1214446559.

[17] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang, "A PDE-based fast local level set method," *J. Comput. Phys.*, vol. 155, no. 2, pp. 410–438, 1999, ISSN: 0021-9991. DOI: 10.1006/jcph.1999.6345. [Online]. Available: https://doi.org/10.1006/jcph.1999.6345.

[18] S. Osher and R. Fedkiw, *Level set methods and dynamic implicit surfaces*, ser. Applied Mathematical Sciences. Springer-Verlag, New York, 2003, vol. 153, pp. xiv+273, ISBN: 0-387-95482-1. DOI: 10.1007/b98879. [Online]. Available: https://doi.org/10.1007/b98879.

[19] J. Sokołowski and J.-P. Zolésio, *Introduction to shape optimization*, ser. Springer Series in Computational Mathematics. Springer-Verlag, Berlin, 1992, vol. 16, pp. ii+250, Shape sensitivity analysis, ISBN: 3-540-54177-2. DOI: 10.1007/978-3-642-58106-9. [Online]. Available: https://doi.org/10.1007/978-3-642-58106-9.

[20] R. Courant, *Dirichlet's Principle, Conformal Mapping, and Minimal Surfaces*. Interscience Publishers, Inc., New York, N.Y., 1950, pp. xiii+330, Appendix by M. Schiffer.

[21] H. Holden and N. H. Risebro, *Front tracking for hyperbolic conservation laws*, Second, ser. Applied Mathematical Sciences. Springer, Heidelberg, 2015, vol. 152, pp. xiv+515, ISBN: 978-3-662-47506-5. DOI: 10.1007/978-3-662-47507-2. [Online]. Available: https://doi.org/10.1007/978-3-662-47507-2.

[22] S. Seifu, personal communication, May 2021.

[23] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd ed. Cambridge University Press, Sep. 2007, ISBN: 0521880688. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20%5C&path=ASIN/0521880688.

Appendix A

Additional Material

Central Difference Discretizations

$$u_x(x_j, y_i, t^n) \simeq \frac{U^n_{i,j+1} - U^n_{i,j-1}}{h_x} \tag{A.1}$$

$$u_y(x_j, y_i, t^n) \simeq \frac{U^n_{i+1,j} - U^n_{i-1,j}}{h_y} \tag{A.2}$$

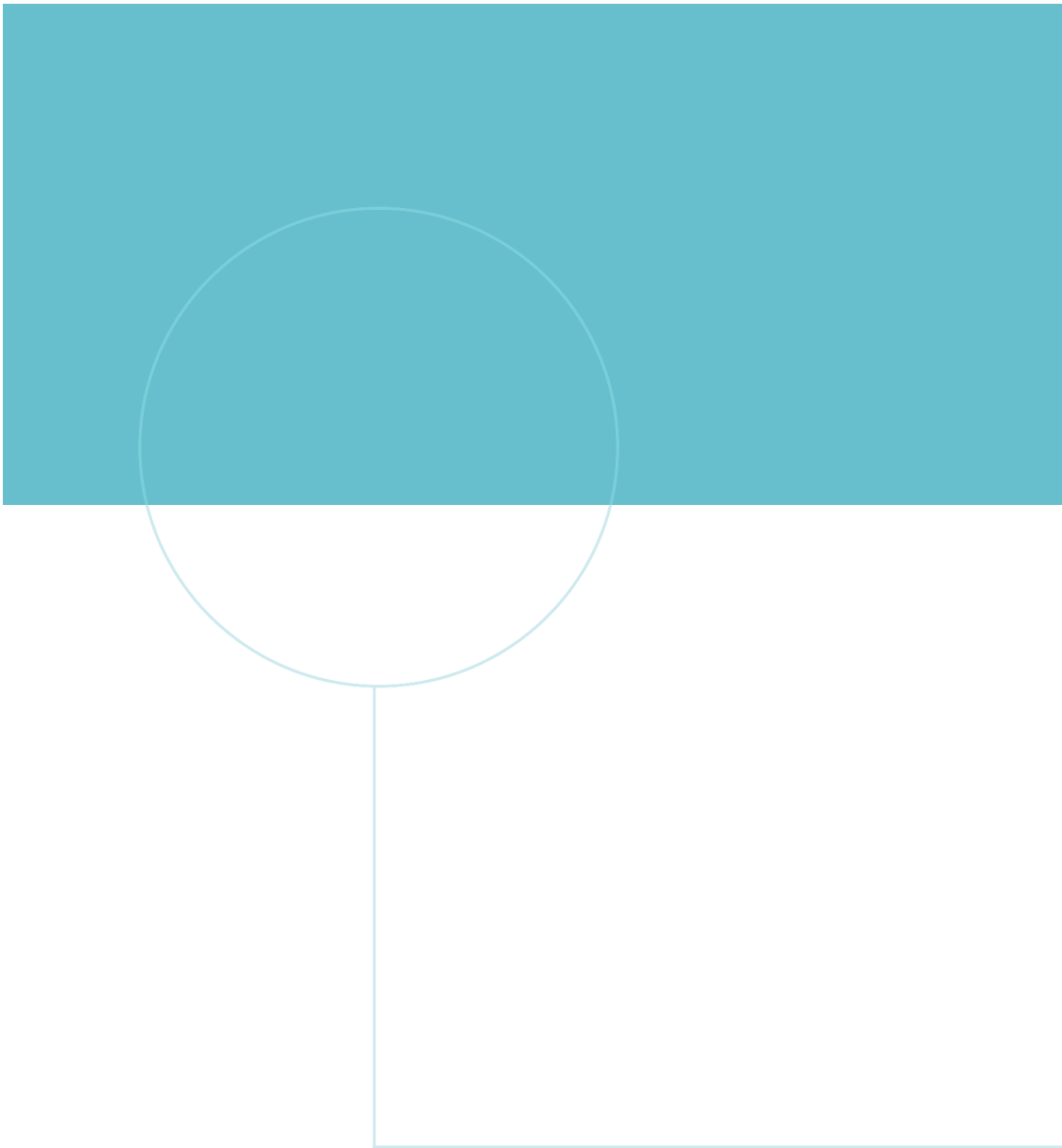
$$u_{xx}(x_j, y_i, t^n) \simeq \frac{U^n_{i,j+1} - 2U^n_{i,j} + U^n_{i,j-1}}{h_x^2} \tag{A.3}$$

$$u_{yy}(x_j, y_i, t^n) \simeq \frac{U^n_{i+1,j} - 2U^n_{i,j} + U^n_{i-1,j}}{h_y^2} \tag{A.4}$$

$$u_{x,y}(x_j, y_i, t^n) \simeq \frac{U^n_{i+1,j+1} - U^n_{i+1,j-1} - U^n_{i-1,j+1} + U^n_{i-1,j-1}}{4h_x h_y} \tag{A.5}$$

Forward Euler

$$u_t(x_j, y_i, t^n) \simeq \frac{U^{n+1}_{i,j} - U^n_{i,j}}{k} \tag{A.6}$$



NTNU

Norwegian University of
Science and Technology