BACHELOR

Applying level set methods to simulate a model for hypha tip-growth

Harcarik, Collin

Award date:
2023

Applying level set methods to simulate a model for hypha tip-growth

Collin Harcarik

July 11, 2023

Contents

Introduction

In an effort to build a more sustainable future, recent advances in building materials have suggested mycelium as a biodegradable insulation material [1]. Mycelium is the typical root-like structure fungi have on and underneath the soil. It consists of a bundle of branching fungal filaments known as hyphae and is the main way by which fungi absorb nutrients from soil.

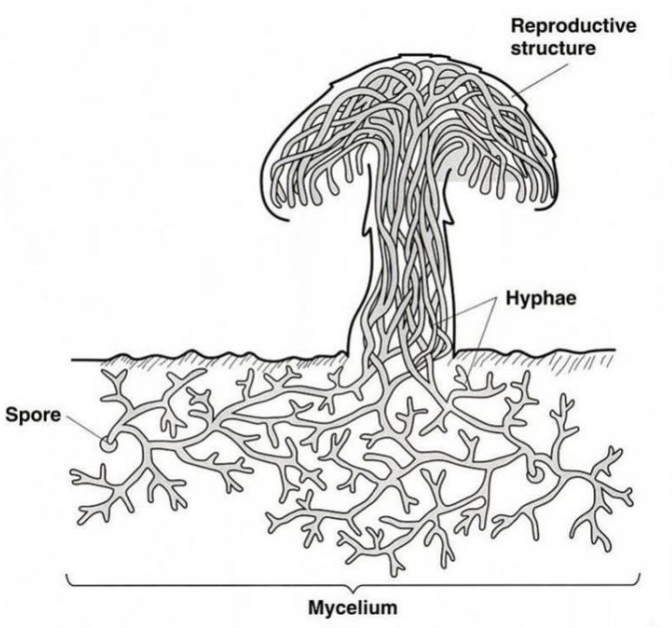


Figure 1: A diagram[1]of the typical anatomy of a fungus

Aside from being useful as a potential building material, the mycelium play an important role in various ecosystems. In terrestrial ecosystems, for instance, the mycelium's growth contributes to the release of carbon dioxide important for the carbon cycling of the ecosystem [2]. Moreover, their decomposition of plant material is vital for enriching the soil with compounds necessary for plant survival. In fact, the mycelium can act as a mediating agent for a plant's nutrient uptake and responses to stress factors such as soil acidification, toxic metals and plant pathogens [3]. As such, it

[1]This diagram was obtained from pinterest: https://www.pinterest.co.uk/pin/383580093238764617/

is not surprising that these mycelial networks can grow to tremendous size. In 1992, the *Armillaria bulbosa* was discovered [4]. Its mycelium covered 15 hectares and at the time was reported to be one of the largest organisms known to man. Since the discovery of *Armillaria bulbosa*, there have been further discoveries of mycelial networks of similar or even larger size [5]. As a result, it has been of increasing interest to study the mechanisms behind the growth of the mycelium. In fact, understanding the growth process and making subsequent predictions may help in realizing mycelial insulation as a sustainable building material and in doing so, building a maintainable future.

In this thesis, we investigate a tip-growth model of the individual fungal filaments, known as hyphae, of the mycelium. In particular, our investigation considers a slight adaptation of the ballistic vesicle supply center model proposed by Bartnicki-Garcie et al. [6] and provides a numerical scheme based on level set methods to simulate said model. In Chapter 1, we will start by providing a brief introduction to the relevant anatomy of the fungal hypha and give a summary of the various models on its growth. In addition, a discussion of the assumptions and relevant mathematical tools will ensue ending ultimately in a derivation of the tethered ballistic vesicle supply center model. Chapter 2 continues with laying out the relevant numerical framework to simulate our model. Correspondingly, the algorithmic approaches such as the fast marching method and narrowbanding are discussed in detail. These techniques are combined into a final algorithm to simulate the tethered ballistic vesicle supply center model. Chapter 3 presents the verification methodology for the more advanced algorithms used and presents the results of these test. After that, the tethered ballistic vesicle supply center model is simulated and the results are shown.

Chapter 1

Mathematical Model

1.1 Hyphal growth

The cells of interest in our model are the branches of the mycelium; that is the fungal filaments formally known as hyphae. These cells can be best described as elongated cylindrical cells with a smooth tip.

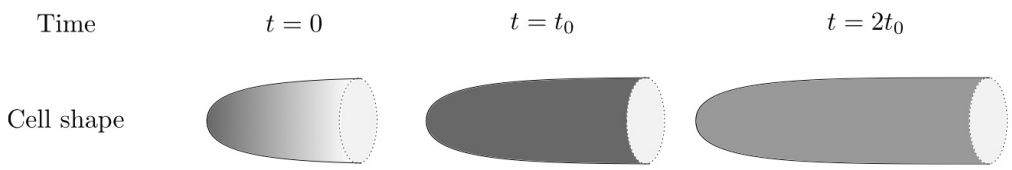


Figure 1.1: Idealized tip growth of a hypha at time steps t_0 [7]

Qualitatively, one can distinguish between two aspects of their morphogenesis, namely the growth in subapical regions; the splitting of a single hyphae into multiple hyphae, and the apical growth [8]. Apical growth originates from the tip; that is the apex of the cell, and is characterized by causing an elongation while maintaining the qualitative shape of the cell. Under the lack of external spatial influences, the hyphal elongation occurs at a constant speed and the cell remains cylindrically symmetric (see Figure 1.1).

Hyphae that exhibit the previously described morphogenesis commonly have an intra-cellular organelle near their tip known as the Spitzenkörper. It is an accumulation of vesicles that, largely, remain near the tip as the cell elongates (see Figure 1.2). In fact, the position of the Spitzenkörper in the cell has been shown to determine the directionality of the hyphal growth [9]. As such, it has been suggested to be the primary mechanism behind hyphal morphogenesis. Although the exact workings of the Spitzenkörper are not known, it is postulated that vesicles are continually produced in this region and subsequently transported to the cell membrane and wall resulting in growth. Importantly, as the hypha grows, the Spitzenkörper remains near the apex of the cell; that is it

moves as the cell grows [9].

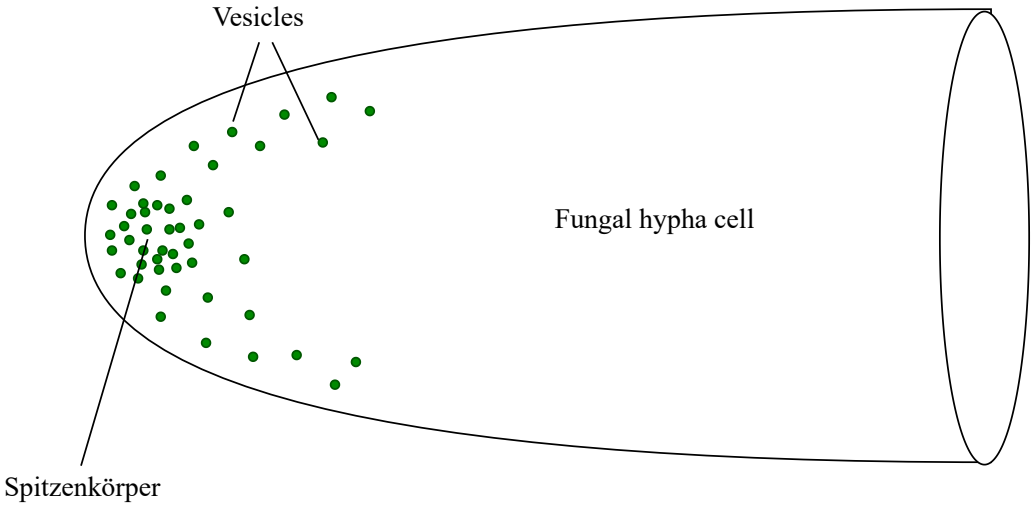


Figure 1.2: The accumulation of vesicles near the tip of a hypha

1.2 Models for hyphal apical growth

In this section, we start by providing a discussion on the models prevalent in the literature. We proceed by introducing the idealization of the hyphal cell that we examine for the remainder of this thesis. Correspondingly, we introduce the three central assumptions made on the growth process within the idealized cell. Finally, these assumptions are combined to produce the tethered ballistic VSC model.

1.2.1 Various models from literature

In the literature, the modelling predominantly focuses on the Spitzenkörper and the material absorption at the cell wall. Regarding material absorption, typical models assume material absorption results in orthogonal growth [10]. The Spitzenkörper, on the other hand, is modelled as a point source called the vesicle supply center (VSC), from which packets of cell wall material, known as vesicles, are emitted at a constant rate. Assuming, moreover, that the VSC is an isotropic moving source from which vesicles travel in straight lines, Bartnicki-Garcie et al. obtained the ballistic VSC model [6]. On the other hand, the vesicles can be assumed to diffuse outwards resulting in Koch's diffusive VSC model [11]. The previous models neglect material properties of the cell wall. In particular, experiments have shown that close to the tip hyphae deform more easily [12]. Campas and Mahadevan modelled this experimental result by the assumption that the cell wall is a thin viscous sheet [13]. In addition, the viscous sheet increases in viscosity as the distance to the apex of the cell is increased. De Jong expanded on this by incorporating the ballistic VSC model and including a model of the hardening of the cell wall with age [7].

In this thesis, the model of interest will be an adapted ballistic VSC model neglecting material properties of the cell wall in which, instead of prescribing the VSC a fixed speed, we consider a VSC tethered to the cell wall.

1.2.2 Idealized hypha

We consider an idealization of a hypha in two dimensions. Typically, hyphae have a cell membrane around which there is a cell wall of non-negligible varying thickness. However, for the purposes of the model, we will consider the cell to have a single enclosing wall of negligible thickness and homogeneous material. In addition, we neglect any influence of biological processes not directly involved in the growth process. These conditions are sufficient for our final model described in Section 1.2.6. However, we will impose an additional condition that our two-dimensional hypha has at least one mirror symmetry about a given line. For one, the symmetry condition will significantly simplify the tether dynamics in Section 1.2.6. Moreover, it is a reasonable simplification, as, in three dimensions, hyphae are approximately rotationally symmetric about an axis during their growth.

Mathematically, we consider an ambient two-dimensional Euclidean space $\mathbb{R}^2$ where we denote by $\langle .,. \rangle : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ the standard inner product. Our idealized cell is modelled as a closed curve $M \subset \mathbb{R}^2$ in this space. Moreover, we assume this curve to be twice differentiable. The reason for this assumption is that we require the curve to have a well-defined curvature. Unsigned[1] curvature is best defined through the concept of an *osculating circle*. In particular, for a sufficiently smooth plane curve ℓ, any point P not at the boundary of the curve can be approximated in the first order by a tangent line and in the second order by an *osculating circle* (see Figure 1.3). The unsigned curvature is simply the inverse of the radius of the *osculating circle*. For our purposes, if we consider the arc-length parametrization, say $x : I \to M$, of our curve M with $I = [0, L]$ and:

$$L = \oint_M dm,$$

then the unsigned curvature $|\kappa_{x(a)}|$ at a point $x(a) \in M$ is given by:

$$|\kappa_{x(a)}| = ||\ddot{x}(a)||.$$

Finally, to incorporate the symmetry condition, we must first define some relevant concepts. In this context, a line l in $\mathbb{R}^2$ is a set of the form:

$$l = \{\vec{x} \in \mathbb{R} : \langle \vec{x}, \vec{a} \rangle = c\},$$

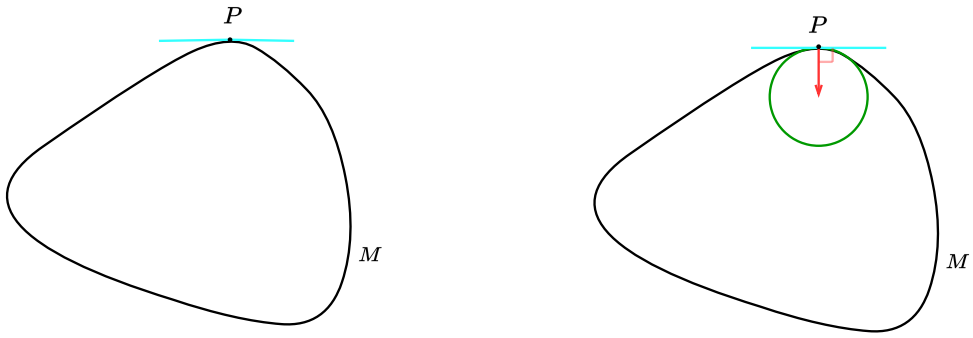
for some vector $\vec{a} \in \mathbb{R}^2$ and scalar $c \in \mathbb{R}$. Using this line l, we can introduce the reflection operator $R_l : \mathbb{R}^2 \to \mathbb{R}^2$ about l defined as:

$$R_l(\vec{v}) = \vec{v} - 2\frac{\langle \vec{v}, \vec{a} \rangle - c}{\langle \vec{a}, \vec{a} \rangle}\vec{a},$$

for $\vec{v} \in \mathbb{R}^2$. Finally, we say a closed curve K is mirror symmetric about a line l if:

$$R_l(K) = K.$$

[1] Our model works with signed curvature, but we refrain from a discussion of orientation of a curve



First order approximation at P Second order approximation at P

Figure 1.3: A tangent line as a first order approximation of our curve M and an osculating circle as a second order approximation of our curve M

Thus, we assume that there is some line, say l, such that our cell M satisfies $R_l(M) = M$. Now, pick $\vec{a} \in \mathbb{R}^2$ and $c \in \mathbb{R}$ such that $l = \{\vec{x} \in \mathbb{R} : \langle \vec{x}, \vec{a} \rangle = c\}$. For the remainder of this thesis, we will consider a convenient coordinate system. In particular, we define a Cartesian coordinate system C' in which $\vec{a} = (0, 1)$ and we translate C' by $-\vec{c} = (0, -c)$ to obtain the coordinate system C, our coordinate system of interest. In this system, we may simplify the reflection operator. Let $\vec{x} \in \mathbb{R}^2$ and denote its coordinates in C by (x_1, x_2), then its reflection in C is given by:

$$T_{-\vec{c}} \circ R_l \circ T_{\vec{c}}(\vec{x}),$$

with $T_{-\vec{c}}$, $T_{\vec{c}}$ a translation by $-\vec{c}$ and $\vec{c}$. In coordinates, we get that:

$$T_{-\vec{c}} \circ R_l \circ T_{\vec{c}}((x_1, x_2)) = T_{-\vec{c}} \circ R_l((x_1, x_2 + c)) = T_{-\vec{c}}((x_1, -x_2 + c)) = (x_1, -x_2),$$

such that the reflection operator $\mathcal{R}_l$ in C is given by:

$$\mathcal{R}_l((x_1, x_2)) = (x_1, -x_2). \tag{1.1}$$

Finally, we will discuss two useful properties of our mirror symmetric cell M in our coordinate system C. In particular, let $\kappa_{\vec{x}}$ be the curvature, $\vec{n}_{\vec{x}}$ be the normal vector at $\vec{x} \in M$ and let $\kappa_{\vec{z}}$ be the curvature, $\vec{n}_{\vec{z}}$ be the normal vector at $\vec{z} = \mathcal{R}_l(\vec{x}) \in M$, then:

$$\kappa_{\vec{x}} = \kappa_{\vec{z}}, \;\; \vec{n}_{\vec{x}} = \mathcal{R}_l(\vec{n}_{\vec{z}}). \tag{1.2}$$

Proof. We will show unsigned curvature preservation explicitly. Denote by $x : I \to M$ the arc-length parametrization of M as defined previously. Note that $\tilde{x} : I \to M$ defined as:

$$\tilde{x} := \mathcal{R}_l \circ x = (x_1, -x_2)$$

is an arc-length parametrization of M, since $\mathcal{R}_l$ is bijective, $\mathcal{R}_l(M) = M$ and:

$$||\dot{\tilde{x}}|| = ||(\dot{x}_1, -\dot{x}_2)|| = ||\dot{x}|| = 1.$$

Now, let $a \in I$, we will show that the unsigned curvature at $x(a)$ is equal to the unsigned curvature at $\tilde{x}(a) = \mathcal{R}_l(x(a))$:

$$|\kappa_{\tilde{x}(a)}| = ||\ddot{\tilde{x}}(a)|| = ||(\ddot{x}_1(a), -\ddot{x}_2(a))|| = ||\ddot{x}(a)|| = |\kappa_{x(a)}|,$$

and this concludes the proof. □

1.2.3 The Spitzenkörper

To model the Spitzenkörper, we make analogous assumptions to those made in the ballistic vesicle supply center model [6]. Particularly, the Spitzenkörper is modelled by a vesicle supply center (VSC) that produces vesicles containing cell wall material at a constant rate. In addition, the VSC is assumed to be a point source that emits its vesicles isotropically. As a result, the VSC physically is producing a vesicle flux field which adds length to the curve. The length being added to the curve is conserved by an evolution of said curve in an underlying balance law, as will be seen explicitly in Section 1.2.6.

Mathematically, we consider the VSC to be at some point, say $\vec{y}$, in $\mathbb{R}^2$ interior to M. Furthermore, we denote the constant rate of vesicles the VSC produces by P. Then, the isotropic vesicle flux field produced by the VSC may be written as a function $\vec{\psi} : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^2$ defined as:

$$\vec{\psi}(\vec{x}, \vec{y}) = \frac{P}{2\pi} \frac{\vec{x} - \vec{y}}{||\vec{x} - \vec{y}||^2}. \tag{1.3}$$

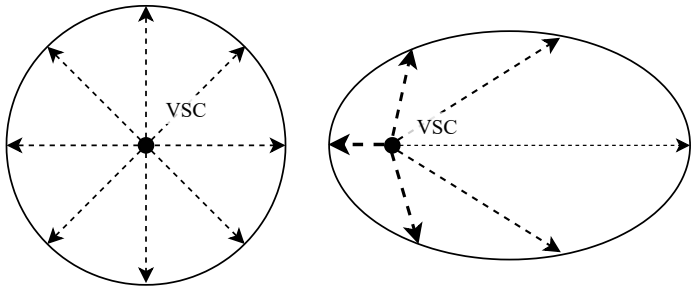


Figure 1.4: The vesicle flux field produced by the VSC. The right picture depicts the inverse proportionality of the vesicle flux field to the distance from the VSC.

The field $\vec{\psi}$ interacts with the cell wall M by adding length to it. In particular, the total length of material absorbed L_{total} along M is given by:

$$L_{\text{total}} = \oint_M \vec{\psi} \cdot d\vec{l}. \tag{1.4}$$

For convenience, we define, in addition, the scalar field $F : M \times \mathbb{R}^2 \to \mathbb{R}$:

$$F(\vec{x}, \vec{y}) = \frac{P}{2\pi} \frac{(\vec{x} - \vec{y}) \cdot \hat{n}}{||\vec{x} - \vec{y}||^2}, \tag{1.5}$$

where $\hat{n}$ is the outward normal vector at $\vec{x}$ on M. This field may physically be interpreted as the material arriving and being incorporated at M per unit of length and time.

1.2.4 Orthogonal growth

As mentioned in Section 1.2.1, *Rhizoctonia solani*, a fungus, was tracked during tip-growth and it was observed that particles on its surface roughly follow orthogonal growth trajectories [10]. However, it seems that not many other fungi have been studied in this context and a particular plant

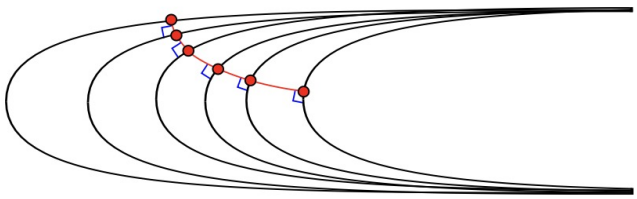


Figure 1.5: Orthogonal growth trajectories of markers on the cell wall [7]

cell, *Medicago truncatula*, has been shown to exhibit tip-growth following non-orthogonal growth trajectories [14]. Regardless, this assumption has been successful in reproducing the characteristic apical morphogenesis of hyphal cells in models such as the ballistic VSC model [6] and therefore, it is reasonable to consider the simplistic assumption that material absorption at the cell wall results in orthogonal growth.

1.2.5 The tether

Another qualitative aspect to the Spitzenkörper's behaviour is its tendency to remain near the tip of the hyphal cell. In [8], Requelme et al. mention a tethering complex responsible for tethering vesicles near the tip for their eventual absorption in the cell membrane. Although they mention that it cannot be definitively said that the tethering complex is essential to apical morphogenesis, we decided to model the mechanism by which the VSC stays near the tip with a tether. In particular, the tether attaches the VSC to some point on the hyphal wall. This differs from the standard ballistic VSC model, as there the VSC is assumed to have a constant velocity. We model the dynamics of the tether using the following assumptions:

1. The tether is rigid, massless and it moves with negligible friction
2. The VSC is subject to friction proportional to is velocity and it moves according to the high friction limit

Mathematically, we denote by $\vec{x}_T \in M$ the tethering point on the cell wall M such that the tether T is an ideal rod connecting $\vec{y}$ and $\vec{x}_T$. We are interested in determining an equation of motion for the VSC; that is for $\vec{y}(t)$ with $t \in [0, \infty)$. For the sake of this discussion, we will assume $\vec{x}_T(t)$ with $t \in [0, \infty)$ to be given. By our first assumption, our tether is rigid and thus, it is of constant

length; that is there is some $D \in \mathbb{R}^+$ such that, $\forall t \in [0, \infty)$:

$$||\vec{x}_T(t) - \vec{y}(t)|| = D. \tag{1.6}$$

Moreover, the tether exerts a force on the VSC and since we disregard frictional effects on the tether, this force will be in the direction of the tether. Therefore, under Newtonian mechanics, the equation of motion for $\vec{y}(t)$ takes the form:

$$\ddot{\vec{y}} = \lambda(\vec{x}_T(t) - \vec{y}(t)) + C(\vec{y}, \dot{\vec{y}}), \tag{1.7}$$

for some $\lambda \in \mathbb{R}$ and some function C. Our next assumption dictates that $C \approx a\dot{\vec{y}}$ with $a \in \mathbb{R}$ such that (1.7) may be simplified:

$$\ddot{\vec{y}} \approx \lambda(\vec{x}_T(t) - \vec{y}(t)) + a\dot{\vec{y}}. \tag{1.8}$$

Finally, we consider (1.8) in the high friction limit, as $a \to \infty$. Although we will not work out the high friction limit analytically, we note that (1.8) is structurally similar to the damped harmonic oscillator for which it is known that, in the high friction limit, the acceleration term drops. As a result, it can be reasonably assumed that in our case the high friction limit corresponds to:

$$\dot{\vec{y}} = \lambda(\vec{x}_T - \vec{y}), \tag{1.9}$$

with $\lambda \in \mathbb{R}$. Finally, we can combine (1.6) and (1.9) to find a λ independent expression for $\dot{\vec{y}}$. In particular, by the constant length assumption, we have that:

$$0 = \frac{d}{dt}||\vec{x}_T - \vec{y}||^2 = 2\langle \vec{x}_T - \vec{y}, \dot{\vec{x}}_T - \dot{\vec{y}} \rangle = 2(\langle \vec{x}_T - \vec{y}, \dot{\vec{x}}_T \rangle - \langle \vec{x}_T - \vec{y}, \dot{\vec{y}} \rangle). \tag{1.10}$$

Now, plugging (1.9) into (1.10), we obtain:

$$0 = \langle \vec{x}_T - \vec{y}, \dot{\vec{x}}_T \rangle - \lambda \langle \vec{x}_T - \vec{y}, \vec{x}_T - \vec{y} \rangle = \langle \vec{x}_T - \vec{y}, \dot{\vec{x}}_T \rangle - \lambda ||\vec{x}_T - \vec{y}||^2,$$

which can be solved for λ:

$$\lambda = \frac{\langle \vec{x}_T - \vec{y}, \dot{\vec{x}}_T \rangle}{||\vec{x}_T - \vec{y}||^2}. \tag{1.11}$$

Using (1.11), we reformulate (1.9) to:

$$\dot{\vec{y}} = \frac{\langle \vec{x}_T - \vec{y}, \dot{\vec{x}}_T \rangle}{||\vec{x}_T - \vec{y}||^2}(\vec{x}_T - \vec{y}). \tag{1.12}$$

Physically, the velocity of the VSC is the projection of the tethering velocity $\dot{\vec{x}}_T$ on the normalized tether $\vec{x}_T - \vec{y}$ (see Figure 1.6).

All in all, our tethering assumptions reduce to conditions (1.6) and (1.12).

1.2.6 Tethered ballistic VSC model

In summary, the tethered ballistic VSC model may be characterized by making the following three assumptions on the growth process within the idealized cell:

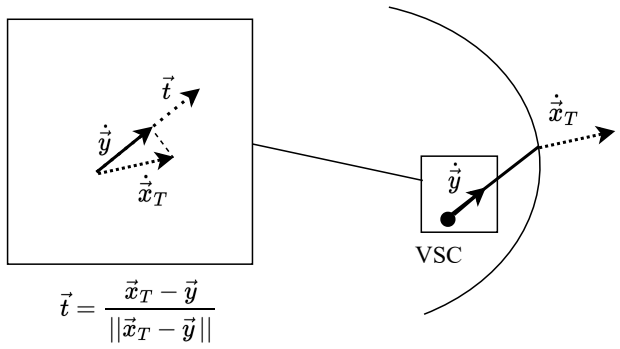


Figure 1.6: The tether and the velocity of the VSC $\dot{\vec{y}}$ in terms of the tether point velocity $\dot{\vec{x}}_T$

1. Material absorption at the cell wall results in **orthogonal growth**,
2. The Spitzenkörper is a VSC that produces vesicles at a **constant** rate and emits them **isotropically** in straight lines,
3. The VSC is **tethered** to a specific point on the cell wall with the corresponding tether **rigid** and satisfying (1.12).

Having discussed the assumptions, we combine them and introduce growth dynamics in a similar fashion to Nolet et al. [15]. In particular, we consider the idealized cell as an evolving one-dimensional closed curve $M(t) \subset \mathbb{R}^2$ with $t \in [0, \infty)$ growing according to a velocity field $\vec{u}(\vec{x}, t)$ defined at $M(t)$. To find $\vec{u}$, we will consider the underlying balance law conserving the length arriving due to the vesicle flux field ψ through an evolution of our cell wall $M(t)$. In particular, using (1.4) and (1.5), we see that for any bounded subset $\Gamma \subset M(t)$ with length $l(t)$, it holds that the length absorbed at Γ per unit time is:

$$\frac{dl}{dt} = \int_\Gamma F \, dl \tag{1.13}$$

In a similar fashion, the change in length of Γ may be related to its geometrical evolution. By the two-dimensional analog of Gauss' formula for the first variation of area, the evolution in terms of the velocity field $\vec{u}$ is:

$$\frac{dl}{dt} = -\int_\Gamma H(\hat{n} \cdot \vec{u}) \, dl, \tag{1.14}$$

where H is the curvature[2], $\hat{n}$ is the outward pointing unit normal to $M(t)$. Since we assume orthogonal growth of the cell, it follows that, for $\vec{x} \in M(t)$, $\vec{u}(\vec{x}, t) = u_n(\vec{x}, t)\hat{n}$. Clearly, $\hat{n} \cdot \hat{n} = 1$ and so, we obtain, by (1.14), that:

$$\frac{dl}{dt} = -\int_\Gamma H u_n \, dl. \tag{1.15}$$

[2]We choose the sign such that $H < 0$ for a circle.

We now combine the flux term (1.13) and evolution term (1.15) to arrive at:

$$-\int_\Gamma Hu_n\,dl = \int_\Gamma F\,dl, \tag{1.16}$$

and, since Γ was arbitrary, it follows that:

$$u_n = -\frac{F}{H}, \tag{1.17}$$

for points in $M(t)$.

We now incorporate the tethering assumption. In Section 1.2.5, we treated the tether in a general setting. However, for the remainder of this thesis, we will consider the tether to be chosen such that it lies on the axis l over which the cell wall M is initially mirror symmetric. This implies that our VSC $\vec{y}$ and tethering point $\vec{x}_T$ lie, at least initially, on the axis l. As mentioned in Section 1.2.2, we consider the coordinate system C for which the axis of symmetry l lies along the horizontal axis and the reflection operator reduces to the form of (1.1). To simplify the expression for $\dot{\vec{y}}$, we note that, by (1.17):

$$\dot{\vec{x}}_T = -\frac{F}{H}\hat{n}. \tag{1.18}$$

with $\hat{n}$ the outward unit normal at $\vec{x}_T$. Since $\vec{x}_T(0)$ lies on l, it follows that $\mathcal{R}_l(\vec{x}_T(0)) = \vec{x}_T(0)$, then, by property (1.2), we have $\mathcal{R}_l(\hat{n}(0)) = \hat{n}(0)$, so $\hat{n}(0) \in l$. In fact, since $\vec{y}(0) \in l$ and $\vec{y}(0)$ lies interior to M, we have that:

$$\hat{n}(0) = \frac{\vec{x}_T(0) - \vec{y}(0)}{||\vec{x}_T(0) - \vec{y}(0)||}. \tag{1.19}$$

Using (1.19), we may simplify (1.12) at $t = 0$ to:

$$\dot{\vec{y}}(0) = \left\langle \frac{\vec{x}_T(0) - \vec{y}(0)}{||\vec{x}_T(0) - \vec{y}(0)||}, \dot{\vec{x}}_T(0) \right\rangle \frac{\vec{x}_T(0) - \vec{y}(0)}{||\vec{x}_T(0) - \vec{y}(0)||} = \dot{\vec{x}}_T(0). \tag{1.20}$$

In Figure 1.7, the setup of our problem in the coordinate system C is sketched.

Extending (1.20) to $t \in (0, \infty)$ requires showing that the evolution of M remains mirror symmetric with respect to l. From our previous discussion, we can frame our evolution problem. In particular, for every point $\vec{x}_0 \in M(0)$ on the initial manifold, we consider the following initial value problem:

$$\begin{cases} \dot{\vec{x}} = \vec{u}(\vec{x}, t, M), \\ \vec{x}(0) = \vec{x}_0, \end{cases} \tag{1.21}$$

and coupled to this, denoting by $\vec{y}_0 \in \mathbb{R}^2$ the initial position of the VSC, we solve the initial value problem:

$$\begin{cases} \dot{\vec{y}} = \dfrac{\langle \vec{x}_T - \vec{y}, \dot{\vec{x}}_T \rangle}{||\vec{x}_T - \vec{y}||^2}(\vec{x}_T - \vec{y}), \\ \vec{y}(0) = \vec{y}_0. \end{cases} \tag{1.22}$$

Solving this entire coupled set of equations will leave us with a solution $M(t)$ for the evolution of the curve. Applying $\mathcal{R}_l$ to $M(t)$ gives us another solution to the evolution problem, since the initial conditions are preserved under the application of $\mathcal{R}_l$. As a result, if we assume the solution $M(t)$

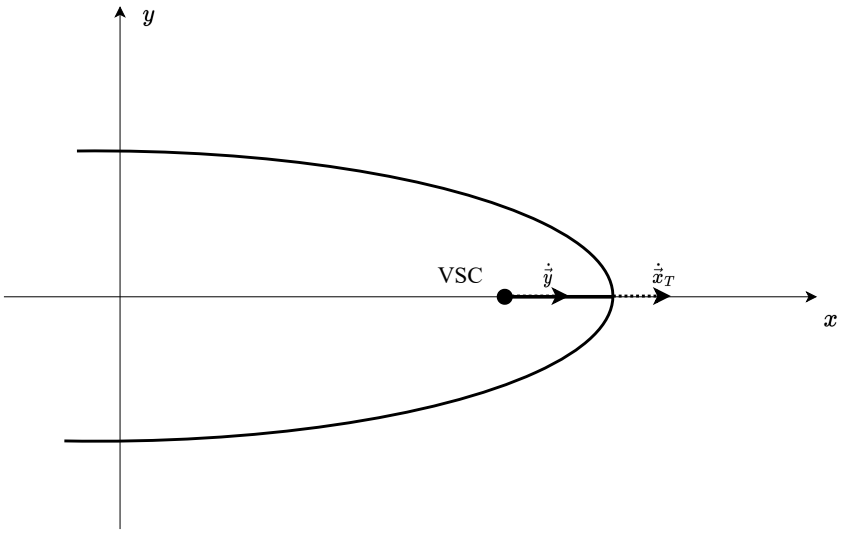


Figure 1.7: The simplifications in our coordinate system C under the symmetry assumption. Note that $\dot{\vec{y}} = \dot{\vec{x}}_T$

to the above evolution problem to be unique, then we must have that $\mathcal{R}_l(M(t)) = M(t)$. Although we will not show uniqueness of solutions to this problem, in the work of van Raaij, local uniqueness and existence of solutions to the 2D ballistic VSC model is shown [16]. It seems reasonable that an analogous result can be derived for our evolution problem. Under this condition, (1.20) can be extended to a time interval $T \subset (0, \infty)$, a property of particular interest to our simulations later on.

Chapter 2

Numerical Scheme

In this chapter, we outline the methodology employed to simulate the tethered ballistic VSC model. In particular, we start by discussing the formulation of the evolution equations. After that, we consider the discretization of our equations providing a basic reasonably stable method to simulate the model. The core method being the fast marching method (FMM). After that, we build upon this method by introducing more efficient and stable techniques, namely narrowbanding and reinitialization. We end this chapter with the final algorithm used to simulate the model.

2.1 Formulation of the evolution equation

As we defined in Section 1.2.6, our cell wall is modelled by an evolving manifold $M(t)$ that grows according to a velocity field $u(\vec{x}, t)$ and one particular way of framing this evolution problem was by the systems (1.21) and (1.22). In practice, this fomulation would require considering a parametrization of the curve and so, it is known as the Lagrangian formulation of the evolution of the curve. Since we are interested in simulating the tethered ballistic VSC model, it is of particular interest how readily this formulation is discretized. The procedure consists of discretizing the parametrization interval and evolving that particular set of points on the curve. Intuitively, *markers* are put on the curve and their evolution is tracked. However, this approach comes with considerable disadvantages. For one, in [17], Sethian notes that curvature approximations of the curve at the *marker* positions are, especially, prone to instability and although a condition on the timestep exists to ensure stability, it is impractical. To elaborate, in this formulation the fixed mesh on the curve results in a discretization that viewed in the ambient space is dynamic (see Figure 2.1) and since the curvature H approximations are independent of the parametrization discretization and rather depend on the ambient discretization $\Delta\vec{h}$, they are extremely sensitive to errors in the positions of the markers. Although there are methods known as front tracking methods to enable a reasonable timestep, they are considerably complex [18]. For a more detailed discussion on these issues, we refer the reader to Section 4.1 in [17].

Alternatively, the front evolution problem can be formulated implicitly. In particular, we consider a function $\phi : \mathbb{R}^2 \times [0, \infty) \to \mathbb{R}$ that satisfies $\{\vec{x} \in \mathbb{R}^2 : \phi(\vec{x}, t) = 0\} = M(t)$ and is twice differentiable on $M(t)$. Moreover, the function ϕ is negative for vectors interior to the curve M and positive for

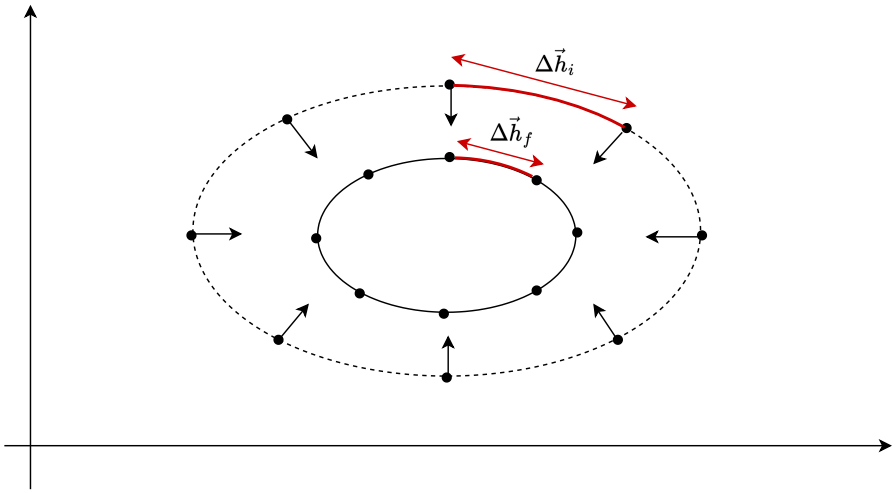


Figure 2.1: The marker method and how the discretization in the ambient space changes, as the curve evolves

vectors exterior to it. This function is commonly referred to as the level set function [18] and it encodes the evolution of our front in its zero level set evolution. The evolution equation can be derived as follows: for $\vec{x} \in M(t)$, we have:

$$\phi(\vec{x}, t) = 0,$$

and differentiating with respect to t yields:

$$\frac{\partial \phi}{\partial t} + \nabla\phi \cdot \vec{u}(\vec{x}, t) = 0, \tag{2.1}$$

where $\nabla\phi$ is the spatial gradient of ϕ. This is known as the level set equation [18] and, along with an initial condition, referred to as the Eulerian formulation of our front propagation problem. Unlike our initial formulation, discretizing (2.1) comes with considerable advantages. For one, it does not require tracking the interface, as its evolution is captured by the evolution of ϕ. Moreover, local geometric properties of the curve are easily determined in terms of the level set function which will be elaborated on in Section 2.1.1. In addition, there are reasonable methods available to ensure the curvature approximations remain stable. The main issue regarding this approach is the added computational overhead. In particular, we evolve ϕ on a 2D grid instead of directly evolving a one-dimensional mesh representing the curve M. However, there are a variety of methods improving the simulation runtime to a point comparable to that of the Lagrangian formulation [18].

2.1.1 Geometric constructs for level set functions

As was shown in 1.2.6, the velocity field $\vec{u}(\vec{x}, t)$ depends both on the curvature H and the unit normal vector of the interface $\hat{n}$. Therefore, it is vital to derive these geometric constructs in terms of the level set function ϕ. For instance, the spatial gradient $\nabla\phi$ of ϕ is perpendicular to

its isocontours and points in the direction of increasing of ϕ. In a similar vein, the curvature at a particular point $\vec{c}$ for the isocontour it lies on can be shown to be:

$$H = -\nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right), \tag{2.2}$$

supposing ϕ is chosen such that $\nabla \phi$ represents the outward pointing normal of the isocontour [18].

These geometric considerations allow us to write out the explicit evolution formulation. We will evolve a level set function $\phi : \mathbb{R}^2 \times [0, \infty) \to \mathbb{R}$ which satisfies $\phi(\vec{x}, 0) = \phi_0(\vec{x})$ where ϕ_0 is twice differentiable on $M(0)$ and $M(0) = \{\vec{x} \in \mathbb{R}^2 : \phi_0(\vec{x}) = 0\}$. For one, we may write out our velocity field (1.17) in terms of ϕ:

$$\vec{u}(\vec{x}, t) = -\frac{F}{H}\hat{n} = \frac{P\left(\vec{x} - \vec{y}\right) \cdot \frac{\nabla \phi}{|\nabla \phi|}}{2\pi ||\vec{x} - \vec{y}\,||^2} \frac{1}{\nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|}\right)} \frac{\nabla \phi}{|\nabla \phi|}. \tag{2.3}$$

Plugging (2.3) into (2.1), we obtain for $\vec{x} \in M(t)$:

$$\frac{\partial \phi}{\partial t} + \frac{P\left(\vec{x} - \vec{y}\right) \cdot \frac{\nabla \phi}{|\nabla \phi|}}{2\pi ||\vec{x} - \vec{y}\,||^2} \frac{1}{\nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|}\right)} |\nabla \phi| = 0. \tag{2.4}$$

Problematically, (2.4) is a statement regarding only $\vec{x} \in M(t)$, since $\vec{u}(\vec{x}, t)$ is only defined at $M(t)$. However, one could automatically extend $\vec{u}(\vec{x}, t)$ such that (2.4) is the evolution equation on all of the domain of ϕ. In fact, we stipulated that ϕ_0 be twice differentiable on its 0 levelset, but this condition can simply be extended to its entire domain. However, the issue lies in the numerical simulation of (2.4). As will be elaborated on in Section 2.2.2, the approximation of curvature (2.2) is likely numerically unstable for all but a special class of level set functions known as signed distance functions. Therefore, we require a velocity extension $\vec{u}_{\text{ext}}$ of $\vec{u}$ to the entirety of $\mathbb{R}^2$ that preserves desirable properties of our level set function ϕ. Since $\vec{u}$ only has a normal component u_n, we will only consider a particular type of velocity extension:

$$\vec{u}_{\text{ext}} = u_{\text{ext}} \frac{\nabla \phi}{|\nabla \phi|}, \tag{2.5}$$

that way we only require a scalar extension u_{ext}. For now, we assume such an extension u_{ext} satisfying $u_{\text{ext}}(\vec{x}, t) = u_n(\vec{x}, t)$ for $\vec{x} \in M$ to be given. So, the evolution problem to be discretized may be written as:

$$\begin{cases} \dfrac{\partial \phi}{\partial t} = -u_{\text{ext}} |\nabla \phi|, \\ \phi(\vec{x}, 0) = \phi_0(\vec{x}). \end{cases} \tag{2.6}$$

2.2 Discretization

In this section, we will discuss the discretization of (2.6). In particular, we will start with the spatial discretization; that is how we approximate the RHS of the evolution equation in (2.6). In discussing this, we will touch upon the main techniques used to keep the curvature approximations stable along with the method used to extrapolate our velocity. Afterwards, we will discuss the temporal discretization and the time stepping used. We will conclude the Section with an overview of how the techniques can be combined into a single algorithm to simulate (2.6).

2.2.1 Spatial domain

We will proceed to define the subset of the domain of ϕ which will be discretized to numerically simulate (2.6) and define some notation used later on. As discussed previously, our cell is modelled as a one-dimensional manifold $M(t)$ in the ambient space $\mathbb{R}^2$. Thus, we are only required to consider a two dimensional plane. As was mentioned in Section 1.2.2, we will consider the coordinate system C in which the axis of symmetry of our curve lies on the horizontal axis. In this system, we choose to consider a rectangular subset of the form $R = [x_L, x_R] \times [y_B, y_T] \subset \mathbb{R}^2$ that suffices to capture the evolution of $M(t)$ for some time interval of interest $[0, T]$. This subset R is then discretized horizontally and vertically using a step size Δh; that is our computational domain D is of the form:

$$D = \{(x_L + i\Delta h, y_B + j\Delta h) : (i, j) \in N\}, \tag{2.7}$$

where $N = \{(i, j) \in \mathbb{N}^2 : (x_L + i\Delta h, y_B + j\Delta h) \in R\}$ is the set of grid coordinates. For convenience, we denote by x_{ij} the point in D associated to the grid coordinate $(i, j) \in N$. We can then naturally introduce notation for functions f defined on our grid D. In particular, we denote by $f_{ij} = f(x_{ij})$ the value of f at the grid point x_{ij}.

2.2.2 The interface velocity

As we mentioned previously, we focus on approximating the RHS of the evolution equation in (2.6). Thus, we consider some fixed time t in our time interval $[0, T]$ and discuss the discretization at that fixed time point. However, we will start by discretizing the interface velocity u_n, as it will be needed to find the extrapolation u_{ext}. By (2.3), we have that for $\vec{x} \in M(t)$:

$$u_n(\vec{x}, t) = \frac{P\left(\vec{x} - \vec{y}\right) \cdot \frac{\nabla\phi}{|\nabla\phi|}}{2\pi||\vec{x} - \vec{y}||^2} \frac{1}{\nabla \cdot \left(\frac{\nabla\phi}{|\nabla\phi|}\right)}.$$

Since our coordinate system C is Cartesian, the normal vector $\nabla\phi$ may be written as:

$$\nabla\phi = \partial_x\phi\, \vec{e}_1 + \partial_y\phi\, \vec{e}_2, \tag{2.8}$$

and, in a similar fashion, the curvature can be written as:

$$\nabla \cdot \left(\frac{\nabla\phi}{|\nabla\phi|}\right) = \frac{(\partial_y\phi)^2\partial_x\partial_x\phi - 2(\partial_y\phi)(\partial_x\phi)\partial_x\partial_y\phi + (\partial_x\phi)^2\partial_y\partial_y\phi}{((\partial_x\phi)^2 + (\partial_y\phi)^2)^{\frac{3}{2}}}. \tag{2.9}$$

To approximate the first and second order derivatives on our grid, we will use central differences:

$$\left(\frac{\partial\phi}{\partial x}\right)_{ij} = \frac{\phi_{(i+1)j} - \phi_{(i-1)j}}{2\Delta h}, \qquad \left(\frac{\partial\phi}{\partial y}\right)_{ij} = \frac{\phi_{i(j+1)} - \phi_{i(j-1)}}{2\Delta h},$$

$$\left(\frac{\partial^2\phi}{\partial x\partial x}\right)_{ij} = \frac{\phi_{(i+1)j} - 2\phi_{ij} + \phi_{(i-1)j}}{\Delta h^2}, \qquad \left(\frac{\partial^2\phi}{\partial y\partial y}\right)_{ij} = \frac{\phi_{i(j+1)} - 2\phi_{ij} + \phi_{i(j-1)}}{\Delta h^2},$$

$$\left(\frac{\partial^2\phi}{\partial x\partial y}\right)_{ij} = \frac{\phi_{(i+1)(j+1)} - \phi_{(i+1)(j-1)} - \phi_{(i-1)(j+1)} + \phi_{(i-1)(j-1)}}{4\Delta h^2}.$$

Note that we do not have to worry about approximating the derivatives at the boundary of D, as we are only approximating the interface velocity of M and the underlying domain R is chosen such as to avoid these complications. These discretizations provide the tools necessary to evaluate the interface velocity u_n on our grid.

Curvature

Although the above approximation seems reasonable, the curvature approximations are highly susceptible to stability issues. In particular, if we consider a general level set function ϕ, as we defined in Section 2.1, then it is very likely that the curvature approximation is numerically unstable. The phenomenon happens in level set functions that experience a flattening and steepening of their spatial gradient around the interface as they evolve. It is known as the *tentpole* phenomenon (see Figure 2.2) and is best resolved by considering a special class of level set functions known as *signed distance* functions [19].

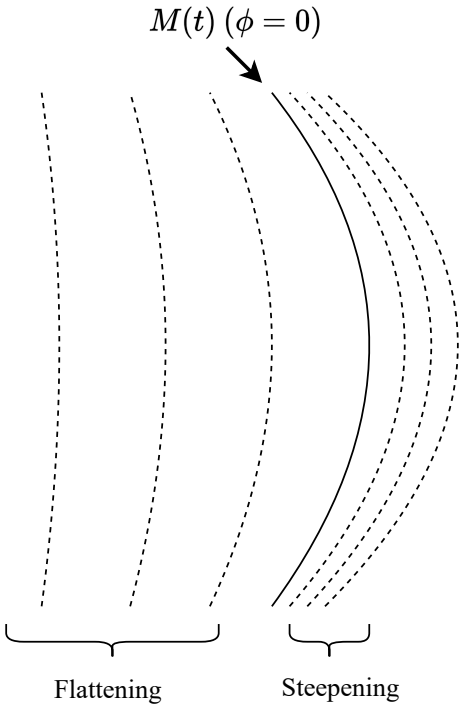


Figure 2.2: A schematic of the *tentpole* phenomenon

As is done in the work of Osher and Fedkiw [18], the signed distance function corresponding to our curve M can be constructed by first considering the *distance* function $d : \mathbb{R}^2 \to \mathbb{R}$ defined as:

$$d(\vec{x}) = \min\{|\vec{x} - \vec{x}_I| : \vec{x}_I \in M\}. \tag{2.10}$$

Then, the signed distance function ϕ is given by d for vectors on and exterior to our curve M and $-d$ for vectors interior to our curve M. Moreover, it can be shown that

$$|\nabla \phi| = 1, \tag{2.11}$$

whenever ϕ is differentiable. This is referred to as the signed distance property [18]. This property ensures that the level set function in differentiable regions does not suffer from the aforementioned

flattening or steepening of the gradient. Ideally, as the level set function is evolved, it retains the signed distance property. In fact, the velocity extension to be discussed in Section 2.2.3 will be chosen such that the property is maintained. With this in mind, we must also alter our initial condition ϕ_0 in (2.6) to satisfy the signed distance property.

2.2.3 The velocity extension

Having discussed the discretization of the interface velocity, we proceed to discuss our choice of velocity extension and the method by which we construct it. In particular, we will follow the work of Adalsteinsson and Sethian [20] in which they address a fast way to construct velocities that maintain the signed distance property. In their paper, the velocity extension of interest is an extension u_{ext} of the normal component velocity at the interface; that is an extension of the form (2.5):

$$\vec{u}_{ext} = u_{ext}\frac{\nabla\phi}{|\nabla\phi|}.$$

As is shown in [20], the necessary condition for u_{ext} to retain the signed distance property of a level set function $\phi(\vec{x}, 0)$ is derived by considering the time derivative of $|\phi|^2$:

$$\frac{\partial|\nabla\phi|^2}{\partial t} = 2\nabla\phi\cdot\frac{\partial\nabla\phi}{\partial t} = 2\nabla\phi\cdot\nabla\frac{\partial\phi}{\partial t}. \tag{2.12}$$

Inserting (2.1), (2.5) into (2.12), we obtain:

$$2\nabla\phi\cdot\nabla\left(-\nabla\phi\cdot u_{\text{ext}}\frac{\nabla\phi}{|\nabla\phi|}\right) = -2\nabla\phi\cdot\nabla(|\nabla\phi|u_{\text{ext}}),$$

such that (2.12) can be written as:

$$\frac{\partial|\nabla\phi|^2}{\partial t} = -2\nabla\phi\cdot\nabla|\nabla\phi|u_{\text{ext}}. \tag{2.13}$$

Clearly, if $\nabla\phi\cdot\nabla u_{\text{ext}} = 0$, then the level set function maintaining the signed distance property (2.11) for all $t\in[0,\infty)$ satisfies (2.13). Moreover, if $\nabla\phi\cdot\nabla u_{\text{ext}} = 0$, the solution to (2.13) can be shown to be unique [20]. It follows that the extension u_{ext} solving the boundary value problem:

$$\begin{cases}\nabla\phi\cdot\nabla u_{\text{ext}} = 0 & \text{on } R,\\ u_{\text{ext}} = u_n & \text{on } M(t),\end{cases} \tag{2.14}$$

is an extension that maintains the signed distance property of a level set function. Intuitively, the extension property can be understood, as the extension being constant on the field lines of $\nabla\phi$ and matching the value of u_n on them (see Figure 2.3).

Before we get to discussing the algorithm to compute u_{ext} according to (2.14), we must consider the problem of computationally solving the boundary value problem:

$$\begin{cases}|\nabla\phi_{\text{temp}}| = 1 & \text{on } R,\\ \phi_{\text{temp}}(\vec{x}) = 0 & \text{for } \vec{x}\in M(t).\end{cases} \tag{2.15}$$

The method, known as the fast marching method (FMM), used to solve (2.15) is the key to solving (2.14) in a computationally efficient manner.

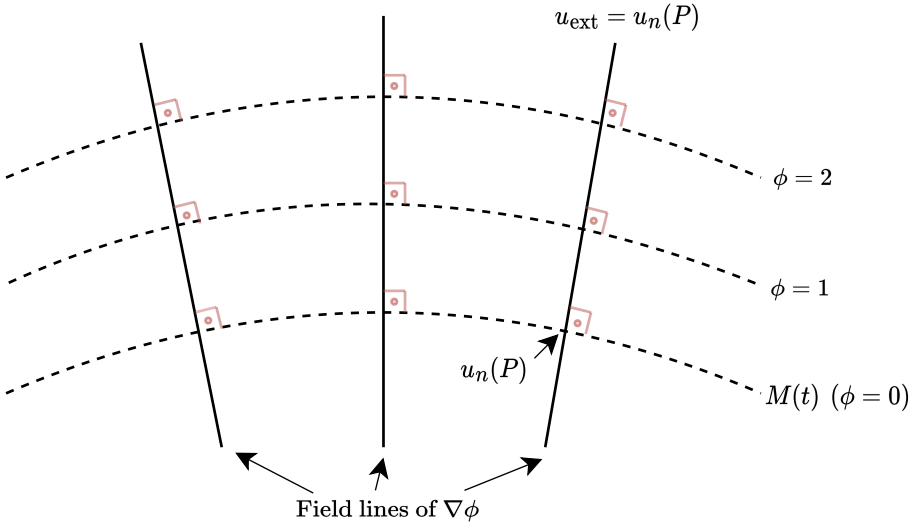


Figure 2.3: The extension velocity u_{ext} is constant on the field lines of $\nabla\phi$ matching the value of u_n on the interface

The Fast Marching Method

We will follow the discussions of the FMM in [20, 21, 22]. Before we get to the algorithm, a numerical approximation of $|\nabla\phi_{\text{temp}}|$ is needed. The approximation needs to correctly deal with corners and cusps in the solution ϕ_{temp}, as it is generally known that (2.15) may become non-differentiable such that a weak solution must be constructed [21]. An appropriate first order scheme that is most convenient for the fast marching method is due to Godunov [21]:

$$\sqrt{\max(D_{ij}^{-x}\phi_{\text{temp}}, -D_{ij}^{+x}\phi_{\text{temp}}, 0)^2 + \max(D_{ij}^{-y}\phi_{\text{temp}}, -D_{ij}^{+y}\phi_{\text{temp}}, 0)^2} = 1, \tag{2.16}$$

with:

$$D_{ij}^{-x}\phi_{\text{temp}} = \frac{(\phi_{\text{temp}})_{ij} - (\phi_{\text{temp}})_{(i-1)j}}{\Delta h}, \qquad D_{ij}^{-y}\phi_{\text{temp}} = \frac{(\phi_{\text{temp}})_{ij} - (\phi_{\text{temp}})_{i(j-1)}}{\Delta h},$$
$$D_{ij}^{+x}\phi_{\text{temp}} = \frac{(\phi_{\text{temp}})_{(i+1)j} - (\phi_{\text{temp}})_{ij}}{\Delta h}, \qquad D_{ij}^{+y}\phi_{\text{temp}} = \frac{(\phi_{\text{temp}})_{i(j+1)} - (\phi_{\text{temp}})_{ij}}{\Delta h}.$$

The core idea to constructing a solution ϕ_{temp} is to exploit the upwind structure in (2.16). In particular, information propagates in a single direction from small values of ϕ_{temp} to larger values and the method exploits this flow to march over the entire grid only once.

We now discuss the method. The problem (2.15) is split into computing ϕ_{temp} interior and exterior to the boundary condition. If the values are to be computed exterior to the boundary condition, the points on our grid interior to the condition are frozen and labeled known. The other points are labeled far. Then, an initial band of values ϕ_{temp} is constructed on our grid using the boundary condition and these points are labeled close points in that they represent the moving front. On every

iteration, the smallest value in the set of close points is frozen and labeled known. The distance to its neighbours is recomputed using (2.16) and updated. In addition, the neighbours not yet in the close point set are labeled close. In that way, the front gradually marches out freezing points that have a set value and adding new points to the moving front (see Figure 2.4). An analogous approach is used to compute the values of ϕ_{temp} interior to the boundary condition.

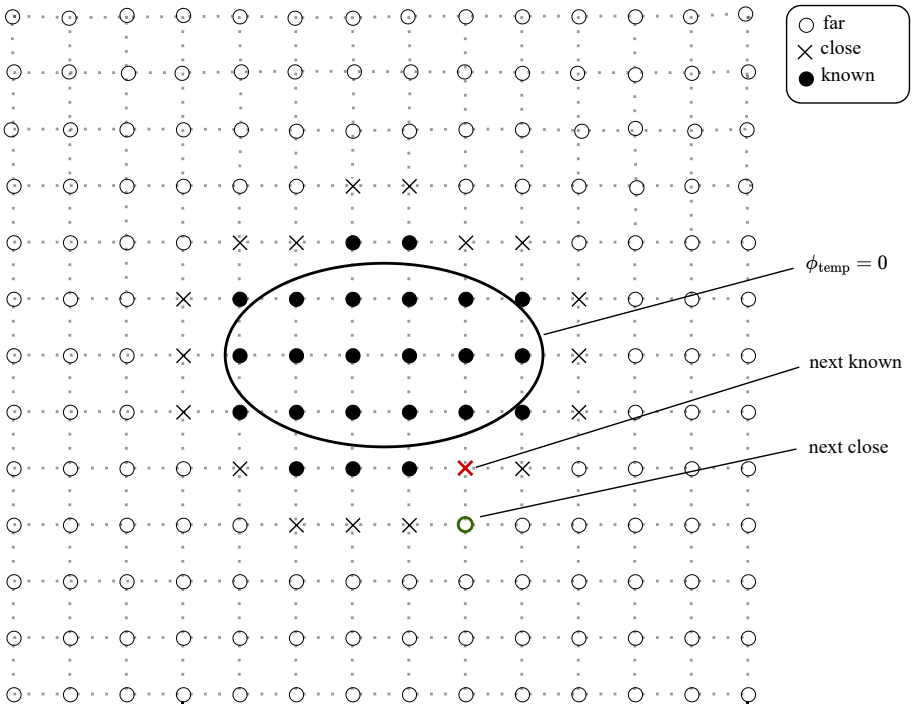


Figure 2.4: A schematic of the FMM marching out from the boundary condition $\phi_{\text{temp}} = 0$

We proceed to describe the procedure in pseudo code (see Algorithm 1) for computing the values of ϕ_{temp} exterior to the boundary condition. Let K, C and F denote three sets for which K contains the known points, C contains the close points and F contains the far points. In addition, $K \cup C \cup F = N$ with initially $F = N, K = \emptyset, C = \emptyset$. Each point in F starts with a value of $\phi_{\text{temp}} = \infty$. In addition, let Neigh : $N \to \mathcal{P}(N)$ denote the function that maps a grid coordinate in N to the set of grid coordinates of its neighbours.

In Lines 11 and 8 of Algorithm 1, we compute the value of $(\phi_{\text{temp}})_{ij}$ for some point $(i, j) \in N$. We will now elaborate on how this is done. Note that (2.16) can be rewritten to:

$$\begin{aligned}\Delta h^2 = \max((\phi_{\text{temp}})_{ij} - (\phi_{\text{temp}})_{(i-1)j}, (\phi_{\text{temp}})_{ij} - (\phi_{\text{temp}})_{(i+1)j}, 0)^2 + \\ \max((\phi_{\text{temp}})_{ij} - (\phi_{\text{temp}})_{i(j-1)}, (\phi_{\text{temp}})_{ij} - (\phi_{\text{temp}})_{i(j+1)}, 0)^2. \end{aligned} \tag{2.17}$$

We consider a single term of (2.17):

$$\max((\phi_{\text{temp}})_{ij} - (\phi_{\text{temp}})_{(i-1)j}, (\phi_{\text{temp}})_{ij} - (\phi_{\text{temp}})_{(i+1)j}, 0)^2. \tag{2.18}$$

Algorithm 1 Fast Marching Method

Require: $F = N, K = \emptyset, C = \emptyset$

Move interior points to the BC from F to K and set their value $\phi_{\text{temp}} = 0$
Initialize C ▷ Details are given in Section 2.2.3
while $C \neq \emptyset$ **do**
    Let (i, j) be the point in C with the smallest value $(\phi_{\text{temp}})_{ij}$
    Move (i, j) to K
    for each $(i_n, j_n) \in \text{Neigh}(i, j)$ **do**
        if $(i_n, j_n) \in F$ **then**
            Compute $(\phi_{\text{temp}})_{i_n j_n}$ using (2.16)
            Move (i_n, j_n) to C
        else if $(i_n, j_n) \in C$ **then**
            Recompute $(\phi_{\text{temp}})_{i_n j_n}$ using (2.16)
            Update $(\phi_{\text{temp}})_{i_n j_n}$ if the recomputed value is smaller than its initial value
        end if
    end for
end while

If $(i-1, j) \notin K$ or $(i+1, j) \notin K$, then its respective value for ϕ_{temp} is treated as ∞ and the maximum value of (2.18) is clear. If $(i-1, j), (i+1, j) \in K$, then one can make the observation that, since $(i, j) \notin K$, we have that:

$$\begin{aligned}
(\phi_{\text{temp}})_{(i-1)j} < (\phi_{\text{temp}})_{(i+1)j} &\implies (\phi_{\text{temp}})_{ij} - (\phi_{\text{temp}})_{(i-1)j} > (\phi_{\text{temp}})_{ij} - (\phi_{\text{temp}})_{(i+1)j}, \\
(\phi_{\text{temp}})_{(i-1)j} > (\phi_{\text{temp}})_{(i+1)j} &\implies (\phi_{\text{temp}})_{ij} - (\phi_{\text{temp}})_{(i-1)j} < (\phi_{\text{temp}})_{ij} - (\phi_{\text{temp}})_{(i+1)j}, \\
&\quad (\phi_{\text{temp}})_{ij} - (\phi_{\text{temp}})_{(i-1)j}, (\phi_{\text{temp}})_{ij} - (\phi_{\text{temp}})_{(i+1)j} \geq 0,
\end{aligned}$$

from which it follows readily what the maximum value of (2.18) is. Having this in mind and since at least one of $(i-1, j), (i+1, j), (i, j-1), (i, j+1)$ is in K, it follows that (2.17) will reduce to a quadratic.

Example 1. *For instance, if all of* $(i-1, j), (i+1, j), (i, j-1), (i, j+1)$ *are in* K*, and* $(\phi_{temp})_{(i-1)j} < (\phi_{temp})_{(i+1)j}$, $(\phi_{temp})_{i(j-1)} < (\phi_{temp})_{i(j+1)}$, *then* (2.17) *reduces to the quadratic:*

$$a[(\phi_{temp})_{ij}]^2 + b(\phi_{temp})_{ij} + c = 0,$$

with:

$$\begin{aligned}
a = 2, \quad b = -2((\phi_{temp})_{(i-1)j} + (\phi_{temp})_{i(j-1)}), \\
c = ([(\phi_{temp})_{(i-1)j}]^2 + [(\phi_{temp})_{i(j-1)}]^2 - \Delta h^2).
\end{aligned}$$

Note that, in general, it can be shown that any quadratic obtained in this way will have at least one real root and in the cases that it has two real roots, we must choose the larger of the two roots to ensure $(\phi_{\text{temp}})_{ij}$ is larger than the other ϕ_{temp} values in the quadratic [22].

We now move on to discussing the runtime of Algorithm 1. The key to this method being fast is choosing a clever data structure for our set C which allows the point with the smallest ϕ_{temp} value

to be found/extracted efficiently and allows other ϕ_{temp} values in the data structure to be updated to smaller values. The ideal data structures for this use case are the heap or binary tree and using them, the run time of this algorithm can quite easily be shown to be $O(|N|\log_2|N|)$. For more information on heaps, we refer the reader to Chapter 6 of [23].

Finally, in our discussion, we focused on a first order Godunov scheme to approximate $|\nabla\phi_{\text{temp}}|$. However, the scheme can be made higher order by replacing the backward difference operators D^{-x}, D^{-y} by a respective higher order backward difference operator and analogously, replacing the forward difference operators D^{+x}, D^{+y} by their higher order counterpart. Note that this change does not affect the runtime of the algorithm, it only changes solving (2.16). For a discussion on solving (2.16) using second order backward and forward differences, we refer the reader to [22]. Note the equations themselves still reduce to quadratics, however, the coefficients are considerably more tedious to derive as the order of the method is increases.

At this point, we have discussed enough details of the FMM to get back to our initial goal to solve (2.14). As is discussed in [20], the idea is to adapt the FMM to simultaneously construct u_{ext} while it is constructing ϕ_{temp} using the velocity extension equation:

$$\nabla\phi_{\text{temp}} \cdot \nabla u_{\text{ext}} = 0. \tag{2.19}$$

Note that the ϕ_{temp} solving (2.15) is, indeed, a signed distance function [18]. In particular, the steps that need to be adapted in the generic FMM (Algorithm 1) are the initialization of C (Line 2) and the computing of values of ϕ_{temp} using (2.16) (Lines 11, 8). To elaborate, the initialization must take care to, also, initialize values for u_{ext} for points in C. Moreover, for Lines 11, 8, after computing the value for $(\phi_{\text{temp}})_{i_n j_n}$, we use it to solve (2.19). In fact, the same difference operators used to find $(\phi_{\text{temp}})_{i_n j_n}$ are used to approximate ∇u_{ext}.

Example 2. *Continuing Example 1, we would approximate* (2.19)*, as follows:*

$$\left(\frac{(\phi_{temp})_{ij} - (\phi_{temp})_{(i-1)j}}{\Delta h}, \frac{(\phi_{temp})_{ij} - (\phi_{temp})_{i(j-1)}}{\Delta h}\right) \cdot \left(\frac{(u_{ext})_{ij} - (u_{ext})_{(i-1)j}}{\Delta h}, \frac{(u_{ext})_{ij} - (u_{ext})_{i(j-1)}}{\Delta h}\right) = 0,$$

which can be simplified to:

$$(u_{ext})_{ij} = \frac{(u_{ext})_{i(j-1)}[(\phi_{temp})_{ij} - (\phi_{temp})_{i(j-1)}] + (u_{ext})_{(i-1)j}[(\phi_{temp})_{ij} - (\phi_{temp})_{(i-1)j}]}{[(\phi_{temp})_{ij} - (\phi_{temp})_{i(j-1)}] + [(\phi_{temp})_{ij} - (\phi_{temp})_{(i-1)j}]}.$$

Initialization of C

To intialize C, we must move all points in F which border K to C. More precisely, for every point $(i,j) \in F$, if:

$$\exists (i_n, j_n) \in \text{Neigh}(i,j) : (i_n, j_n) \in K,$$

we move (i,j) from F to C. After this, we must initialize the ϕ_{temp} values at all points in C. To this end, for each point $(i,j) \in C$ we consider the neighbours $K_{(i,j)} \subset \text{Neigh}(i,j)$ that are known and for each $(i_n, j_n) \in K_{(i,j)}$ we determine the distance of the front to (i,j) on the grid line between (i,j) and (i_n, j_n).

Example 3. *For instance, consider $(i,j) \in C$ and suppose $K_{(i,j)} = \{(i+1,j),(i,j-1)\}$. Then, we determine the distance s from the point at (i,j) to the front on the grid line from (i,j) to $(i,j-1)$ using linear*[1] *interpolation of the inputted level set function ϕ:*

$$s = \frac{|\phi_{ij}|\Delta h}{|\phi_{ij}| + |\phi_{i(j-1)}|},$$

and similarly, for the distance t on the grid line from (i,j) to $(i+1,j)$:

$$t = \frac{|\phi_{ij}|\Delta h}{|\phi_{ij}| + |\phi_{(i+1)j}|},$$

The situation is schematically shown in Figure 2.5.

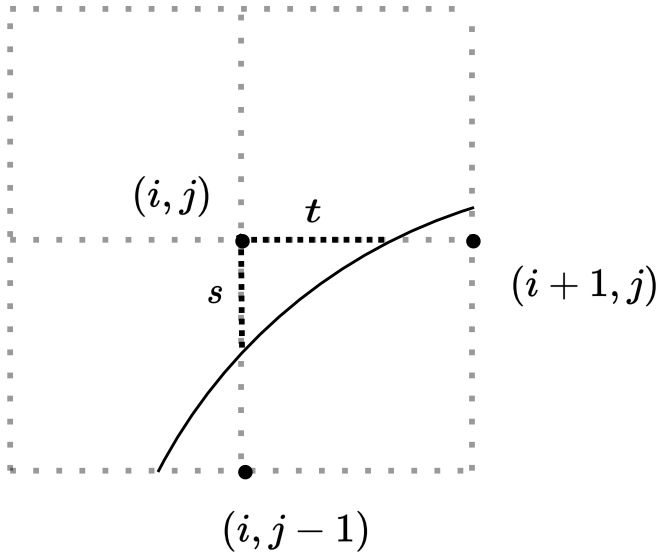


Figure 2.5: An example of the distances s, t computed during the initialization of C

These distances are then combined to obtain a ϕ_{temp} value for the point at (i,j). In Section 4.1 of [20], Adalsteinsson and Sethian provide an excellent explanation of how this is done in all the different cases, so we refer the reader to this discussion for those additional details. However, the initialization must also initialize the scalar velocity u_{ext} at points in C. This is done in a similar fashion to how the distances are initialized; that is for each $(i,j) \in C$ we consider $K_{(i,j)}$ as defined above and for each $(i_n, j_n) \in K_{(i,j)}$, we determine the velocity at the front on the grid line between (i,j) and (i_n, j_n) by linearly interpolating $(u_n)_{ij}$ and $(u_n)_{i_n j_n}$.

Example 4. *Continuing example 3, the velocity u^s_{ext} at the front on the grid line between (i,j) to $(i,j-1)$ is:*

$$u^s_{ext} = \frac{|\phi_{ij}|}{|\phi_{ij}| + |\phi_{i(j-1)}|}(u_n)_{i(j-1)} + \frac{|\phi_{i(j-1)}|}{|\phi_{ij}| + |\phi_{i(j-1)}|}(u_n)_{ij}.$$

[1] Higher order FMMs would have to use a higher order technique

Similarly, the velocity u^t_{ext} at the front on the grid line between (i,j) to $(i+1,j)$ is:

$$u^t_{ext} = \frac{|\phi_{ij}|}{|\phi_{ij}| + |\phi_{(i+1)j}|}(u_n)_{(i+1)j} + \frac{|\phi_{(i+1)j}|}{|\phi_{ij}| + |\phi_{(i+1)j}|}(u_n)_{ij}.$$

These velocities are then combined to obtain a u_{ext} value for the point at (i,j). Once again, Adalsteinsson and Sethian provide an excellent explanation of this step and we refer the reader to Section 4.2 of [20] for the remaining details.

2.2.4 Dealing with the boundary of our grid

We have, now, described the complete procedure to approximating u_{ext}. The remaining factor of the RHS of (2.6) is $|\nabla\phi|$. Since we approximated the interface velocity u_n using central differences, we apply the same central differences to approximate $|\nabla\phi|$. However, unlike u_n, we must approximate $|\nabla\phi|$ at the boundary of our grid. For this special case, we use the second order backward and forward differences:

$$\left(\frac{\partial\phi}{\partial x}\right)_{ij} = \frac{3\phi_{ij} - 4\phi_{(i-1)j} + \phi_{(i-2)j}}{2\Delta h}, \qquad \left(\frac{\partial\phi}{\partial x}\right)_{ij} = \frac{-3\phi_{ij} + 4\phi_{(i+1)j} - \phi_{(i+2)j}}{2\Delta h}.$$

Analagous formulas can be derived for the partial derivative with respect to y.

2.2.5 Time discretization

In the preceding sections, we discussed how our level set function ϕ at a fixed time point $t \in [0,T]$ is spatially discretized and how we approximate the RHS of the evolution equation in (2.6). In a similar fashion, we must discretize the time interval $[0,T]$ over which we would like to evolve ϕ. However, contrary to the spatial discretization, we cannot choose a fixed time step and it must be chosen dynamically as the front is evolving. To elaborate, we discretize the time interval $[0,T]$ as a set of times $0 = t_0 < t_1 < \cdots < t_m < T$ which we denote by E. However, we require that the differences $\Delta t_n = t_{n+1} - t_n$ for $0 \le n \le m-1$ are chosen such that:

$$\Delta t_n < \frac{\Delta h^2}{\max\{|\vec{u}_{\text{ext}}(\vec{x},t_n)| : \vec{x} \in D\}}. \tag{2.20}$$

This is known as the CFL condition and it is a necessary condition for our numerical scheme of (2.6) to be stable [18]. In our present context[2], we used central differences for the approximation of the interface velocity u_n, $|\nabla\phi|$ and therefore, our condition contains the factor Δh^2. Note that the condition (2.20) is time explicit and therefore, the step size Δt_i can be chosen iteratively in the final algorithm.

For functions g defined on E, we denote by $g^n = g(t_n)$ for $t_n \in E$. Using this, we may discuss our time integration methods. The basic method we use is the first order explicit forward Euler method:

$$\left(\frac{\partial\phi}{\partial t}\right)^n = \frac{\phi^{n+1} - \phi^n}{\Delta t_n}.$$

[2]For more details see Section 3.2 in [18]

In our context, the forward Euler step looks as follows:

$$\phi_{ij}^{n+1} = \phi_{ij}^{n} - \Delta t_n (u_{\text{ext}})_{ij}^{n} (|\nabla \phi|)_{ij}^{n},$$

for the levelset function.

2.2.6 Basic level set method

We are now at a point at which we can describe a basic level set method to numerically solve (2.6). The method is described in pseudocode in Algorithm 2.

Algorithm 2 Basic Level Set Method

1: Initialize ϕ_{ij}^0 to $(\phi_0)_{ij}$ for all $(i,j) \in N$
2: **for** $k \leftarrow 0$ to $m-1$ **do**
3: Compute u_n near the 0 level set of ϕ^k on N
4: Find the velocity extension u_{ext} on our grid using Algorithm 1
5: Compute $(|\nabla \phi|)_{ij}^k$ for each $(i,j) \in N$
6: Choose Δt_k according to (2.20)
7: Evolve ϕ_{ij}^k to ϕ_{ij}^{k+1} using a forward Euler timestep for each $(i,j) \in N$
8: Evolve the VSC $(\vec{y})^k$ to $(\vec{y})^{k+1}$ using a forward Euler time step and the velocity u_n at the tethering point $\vec{x}_T$
9: **end for**

2.3 Efficient level set method

In the previous sections, we provided a discretization and scheme for numerically solving (2.6). The scheme is stable enough to produce results, but suffers from two issues. In general, the grid N over which the level set function ϕ is evolved is large if we want to evolve it to a reasonable time T and therefore, the basic level set method will be very slow. Additionally, the stability properties of the basic scheme were experimentally not ideal. Therefore, we take the time to improve upon the basic technique using two methods from the level set literature, namely reinitialization and narrowbanding [18].

2.3.1 Reinitialization

The goal of reinitialization is to take an arbitrary level set function ψ and reinitialize it to a signed distance function ϕ that has the same 0 level set. Note that this goal is in fact equivalent to solving (2.15) for ϕ using the FMM. However, there is a major downside to using the FMM as the method of reinitialization. The issue is that higher order FMM are increasingly difficult to implement. To elaborate, initializing the FMM requires interpolation and as the order is increased, this step requires the use of root finding algorithms. To avoid these complications, we opted to implement another method that at the cost of speed was considerably less complex.

We will follow the discussion of Osher and Fedkiw in Section 7.4 of [18]. Referring to the context we set up in Section 2.2, we have a level set function ϕ at some time point $t_n \in E$ and we would like to solve:

$$\begin{cases} \dfrac{\partial \phi_{\text{reinit}}}{\partial \tau} + S(\phi(\cdot, t_n))(|\nabla \phi_{\text{reinit}}| - 1) = 0, \\ \phi_{\text{reinit}}(\cdot, 0) = \phi(\cdot, t_n), \end{cases} \tag{2.21}$$

to equilibrium in τ with:

$$S(\phi(\vec{x}, t_n)) = \begin{cases} 0 & \text{if } \phi(\vec{x}, t_n) = 0, \\ 1 & \text{if } \phi(\vec{x}, t_n) > 0, \\ -1 & \text{if } \phi(\vec{x}, t_n) < 0, \end{cases}$$

for $\vec{x} \in R$. Then, we may use the solution ϕ_{reinit} at equilibrium to reinitialize $\phi(\cdot, t_n)$ to a signed distance function. The above equation in (2.21) is known as the *reinitialization* equation.

Now, the reinitialization equation (2.21) is a nonlinear hyperbolic equation. Osher and Fedkiw recommend spatially discretizing the hyperbolic term $S(\phi(\cdot, t_n))|\nabla \phi_{\text{reinit}}|$ using the smeared sign function S which on our grid N is:

$$S_{ij} = \frac{\phi_{ij}^n}{\sqrt{(\phi_{ij}^n)^2 + \Delta h^2}}. \tag{2.22}$$

For the factor $|\nabla \phi_{\text{reinit}}|$, a Godunov scheme with ENO upwinding is used. In particular, on our grid, the Godunov scheme can be shown to reduce to [18]:

$$\left[\left(\frac{\partial \phi_{\text{reinit}}}{\partial x}\right)^2\right]_{ij} = \begin{cases} \max(\max(D_{ij}^{-x}\phi_{\text{reinit}}, 0)^2, \min(D_{ij}^{+x}\phi_{\text{reinit}}, 0)^2) & \text{if } S_{ij} > 0, \\ \max(\min(D_{ij}^{-x}\phi_{\text{reinit}}, 0)^2, \max(D_{ij}^{+x}\phi_{\text{reinit}}, 0)^2) & \text{if } S_{ij} < 0, \end{cases} \tag{2.23}$$

with D^{-x} and D^{+x} the backward and forward ENO difference respectively.

ENO differencing

ENO stands for essentially non-oscillatory and is a scheme commonly used for hyperbolic partial differential equations. To elaborate, solutions to hyperbolic PDEs are known to have sharp gradients or even discontinuities such that a fixed stencil finite difference is ill-advised. This is due to the stencil potentially containing the discontinuity which causes oscillations at points using this stencil. Therefore, ENO schemes use adaptive stencils to avoid including discontinuities and as a result, limit oscillations. Another aspect to ENO schemes is that they extend simple first order upwinding, the simple backward and forward difference, to higher orders. We will not dive into the details of deriving these schemes. We refer the reader to Chapter 5 of [24]. However, we must make an important note regarding the use of ENO schemes near the boundary of our grid. In particular, any points that are required to determine the different Newton interpolants are extrapolated linearly. For instance, suppose we determine the second order backward difference of ϕ_{reinit} at the boundary point $(1,1) \in N$. Then, possible stencils for $D_{(1,1)}^{-x}\phi_{\text{reinit}}$ are $S_1 = \{(\phi_{\text{reinit}})_{(-1,1)}, (\phi_{\text{reinit}})_{(0,1)}, (\phi_{\text{reinit}})_{(1,1)}\}$ and $S_2 = \{(\phi_{\text{reinit}})_{(0,1)}, (\phi_{\text{reinit}})_{(1,1)}, (\phi_{\text{reinit}})_{(2,1)}\}$. However, neither $(\phi_{\text{reinit}})_{(0,1)}$ nor $(\phi_{\text{reinit}})_{(-1,1)}$ exist on our grid, so we interpolate $(\phi_{\text{reinit}})_{(1,1)}$ and $(\phi_{\text{reinit}})_{(2,1)}$ linearly to extrapolate values of $(\phi_{\text{reinit}})_{(0,1)}, (\phi_{\text{reinit}})_{(-1,1)}$. The ENO difference is then

found as it is usually determined. This procedure works as long as the point at which we would like to determine the ENO differences has at least one neighbour with which it can be interpolated.

This concludes the spatial discretization of (2.21). For the time discretization, an explicit total variation diminishing (TVD) Runge-Kutta timestep of the same or higher order than the spatial scheme should be used [18]. In our case, we used an explicit fourth order TVD Runge-Kutta method[3] (see Section 3.5 in [18]), even though we restricted ourselves to third order ENO schemes. Finally, the timestep $\Delta\tau$ during each iteration is chosen such that the CFL condition is satisfied:

$$\Delta\tau < \Delta h.$$

2.3.2 Narrowbanding

In [25], Adalsteinsson and Sethian discuss a level set method with a significant speed up over the basic method. In particular, the idea is that we restrict the evolution of our level set function ϕ to a narrowband about its 0 level set, since this is the front of interest. As will be elaborated later on, the implementation of this approach comes with considerable complexity, but the method is particularly advantageous to our problem. For one, since our interface of interest is one-dimensional, it can be represented by $O(\sqrt{|N|})$ points on our grid. So, a narrowband approach that considers a width of w cells about the interface will lead to a method that updates $O(w\sqrt{|N|})$, a considerable improvement to $O(|N|)$ given w is not too large. Another advantage is that we may restrict our velocity extension to this narrowband instead of computing it on the entire grid. Thus, evolving ϕ by a single step will take $O(w\sqrt{|N|}\log_2(w\sqrt{|N|}))$ instead of the basic implementation which takes $O(|N|\log_2|N|)$.

The basic approach is that starting from a level set function ϕ given on our grid, we find the narrowband around the 0 levelset and label the points within the narrowband as active and those exterior faraway. In addition, the points within the narrowband that neighbour faraway points are labeled boundary (see Figure 2.6). Now, the level set function ϕ is evolved until its 0 level set reaches one of boundary points at which point the narrowband is rebuilt around the 0 level set again. We proceed with a generic description of the method in pseudo code (see Algorithm 3). Let A, B and F denote three sets for which A is the set of active points, B contains the boundary points and F the faraway points. Moreover, let $\text{Neigh} : N \to \mathcal{P}(N)$ denote the function that maps a grid coordinate in N to the set of grid coordinates of its neighbours. Our method slightly differs from that described in [25], as we rebuild the narrowband by reinitializing ϕ over the entire grid instead of the more efficient approach described in [25]. However, given the sporadic use of reinitialization steps, the additional runtime associated with this choice does not significantly affect the extreme speed up achieved over Algorithm 2.

Although initializing the narrowband is pretty straightforward due to ϕ being a signed distance function, there is a particular issue that needs to be considered. The points that lie within the narrowband must have sufficient neighbours to compute the finite difference approximations of the derivatives. In particular, we will require that each point within the narrowband has at least one horizontal and vertical neighbour that also lies within the narrowband. We will refer to achieving this property of the narrowband as smoothing, although it should not be confused with the smoothing applied in [25].

[3] We will provide some details on this method in Section 2.3.3

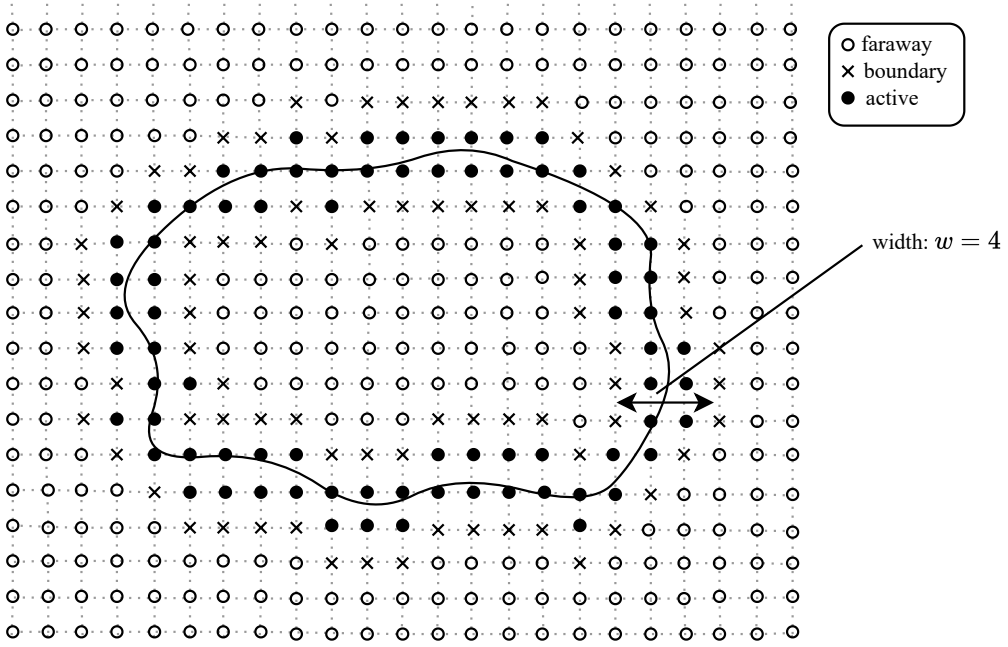


Figure 2.6: An example of a narrowband around a front

Smoothing

Before we dive into the smoothing, we must discuss the cell width w of the narrowband. In particular, as is noted in [25], a small width w of 6 or less leads to frequent reinitialization and significatly hampers the runtime improvements achieved by the narrowbanding technique. On the other hand, a large narrowband will increase the runtime of the time stepping within the narrowband. Therefore, Sethian mentions that a size of about $w = 12$, an initial narrowband having about a width of 6 cells on either side of the 0 level set, is a reasonable balance between number reinitialization steps and runtime of computations within the narrowband [25]. With that in mind, we kept our choices of $w \geq 10$.

Algorithm 3 Narrowbanding

Require: $F = N, A = \emptyset, B = \emptyset$

1: Build the narrowband A and B
2: **for** $k \leftarrow 0$ to $m - 1$ **do**
3: Evolve ϕ^k to ϕ^{k+1}
4: **if** $\exists\,(i,j) \in B, \exists\,(i_n, j_n) \in \text{Neigh}(i,j) : \text{sign}(\phi_{ij}^{k+1}) \neq \text{sign}(\phi_{i_n j_n}^{k+1})$ **then**
5: Reinitialize ϕ^{k+1} to a signed distance function on the entire grid
6: Rebuild the narrowband A and B
7: **end if**
8: **end for**

For convenience, we define N_{nb} to be the narrowband $B \cup A$. We say a single point $(i, j) \in N$ on our grid satisfies the smoothing property with respect to the narrowband N_{nb} if smooth : $N \to \{true, false\}$ defined as:

$$\text{smooth}(i, j) := [(i + 1, j) \in N_{nb} \vee (i - 1, j) \in N_{nb}] \wedge [(i, j + 1) \in N_{nb} \vee (i, j - 1) \in N_{nb}],$$

evaluates to *true* at (i, j). Thus, our narrowband N_{nb} is said to be smooth if:

$$\forall (i, j) \in N_{nb} : \text{smooth}(i, j). \tag{2.24}$$

In general, when constructing the narrowband using the signed distance function ϕ, the narrowband may have $(i, j) \in B$ for which $\text{smooth}(i, j) = false$ (see Figure 2.7). Therefore, after building N_{nb} using the signed distance function ϕ, we iterate through N_{nb} and for each $(i, j) \in N_{nb}$ with $\text{smooth}(i, j) = false$, we pick $(i_n, j_n) \in \text{Neigh}(i, j) \setminus N_{nb}$ with $\text{smooth}(i_n, j_n) = true$ such that upon adding (i_n, j_n) to N_{nb}, we have that $\text{smooth}(i, j) = true$. These neighbouring points are then added to N_{nb} and the boundary B, active points A are updated accordingly. Although we will refrain from proofing this procedure works, we note that, in practice using this building method, we have always obtained a narrowband N_{nb} satisfying (2.24) with at most 10 points being added[4].

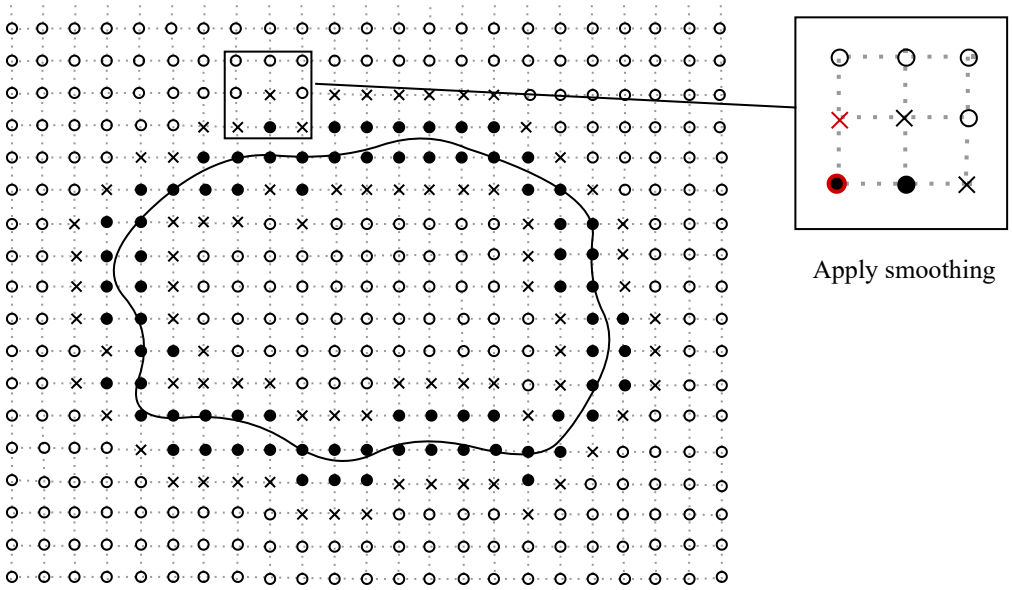


Figure 2.7: An example of our smoothing procedure

The entire point behind the smoothing procedure is that we can now apply ENO differencing everywhere on our narrowband N_{nb}, as there will always be a neighouring point to linearly extrapolate the stencil points needed to construct second or third order ENO differences. Now, we can discuss the discretization of the advection equation in (2.6) on our narrowband. The interface velocity u_n can still be approximated as it was in Section 2.2.2, since the narrowband is rebuilt whenever the interface hits the boundary. Moreover, the FMM requires no adaptation to run on the narrowband

[4]This is to say that the cost of this procedure with regards to the runtime is negligible

and so, the entire velocity construction u_{ext} remains unchanged on the narrowband. To approximate the $|\nabla\phi|$ factor, we change our scheme to use a Godunov scheme with second order ENO upwinding. For instance, at $(i,j) \in N_{nb}$, we employ:

$$\left[\left(\frac{\partial\phi}{\partial x}\right)^2\right]_{ij} = \begin{cases} \max(\max(D_{ij}^{-x}\phi, 0)^2, \min(D_{ij}^{+x}\phi, 0)^2) & \text{if } (u_{\text{ext}})_{ij} > 0, \\ \max(\min(D_{ij}^{-x}\phi, 0)^2, \max(D_{ij}^{+x}\phi, 0)^2) & \text{if } (u_{\text{ext}})_{ij} < 0, \end{cases}$$

with D^{-x} and D^{+x} the backward and forward ENO difference respectively. This concludes the spatial discretization of the advection equation in (2.6) on our narrowband.

Finally, another benefit to our smooth narrowband is that we can run the reinitialization equation (2.21) on the narrowband itself; that is we reinitialize ϕ to a signed distance on the narrowband. Note that, although this is not necessary since ϕ remains close to a signed distance function under advection by u_{ext}, we see in our simulations later on (Section 3.1.2) that solving (2.21) until $\tau \geq w/2$ on our narrowband after advancing ϕ a single time step improves the accuracy of the method. Moreover, this comes at a small runtime cost, since (2.21) is only run on the narrowband and we only solve until $\tau \geq w/2$.

2.3.3 Runge-Kutta time stepping

Using narrowbanding, our method will become very rapid and so, we examined higher order time schemes for (2.6). Since we use a Godunov scheme with ENO upwinding for the $|\nabla\phi|$ factor in the advection equation (2.6), a well-suited class of time schemes to use in conjunction with this spatial discretization are the explicit total variation diminishing (TVD) Runge-Kutta schemes [18]. We will discuss how to apply the second order TVD Runge-Kutta in our context explicitly. Note that higher order methods can be found in Section 3.5 of [18] and can be performed algorithmicly in a similar fashion. Algorithm 4 depicts evolving ϕ^k to ϕ^{k+1} on our narrowband.

2.3.4 Bringing it all together

We proceed to combine the above techniques into a single efficient method to solve (2.6) numerically (see Algorithm 5). Note that Algorithm 5 represents an explicit version of Algorithm 3.

Algorithm 4 Second Order TVD Runge-Kutta

Require: ϕ^k and VSC $\vec{y}^{\,k}$ is given

Compute $(u_n)^k$ near the interface

Find $(u_{\text{ext}})^k$ using the FMM on our narrowband

Find $|\nabla\phi|^k$ on our narrowband

Choose Δt_k according to (2.20) (CFL condition)

Perform a forward Euler step; for each $(i,j) \in N_{nb}$:
$\tilde{\phi}_{ij}^{k+1} \leftarrow \phi_{ij}^k - \Delta t_k (u_{\text{ext}})_{ij}^k |\nabla\phi|_{ij}^k$

Evolve the VSC $\vec{y}^{\,k}$ to $\vec{y}^{\,k+1}$ using a forward Euler time step and the velocity $(u_n)^k$ at the tethering point $\vec{x}_T$

Compute $(\tilde{u}_n)^{k+1}$ near the interface ($\tilde{\phi}^{k+1}$ and $\vec{y}^{\,k+1}$ are used)

Find $(\tilde{u}_{\text{ext}})^{k+1}$ using the FMM on our narrowband

Find $|\nabla\tilde{\phi}|^{k+1}$ on our narrowband

Perform a forward Euler step; for each $(i,j) \in N_{nb}$:
$\tilde{\phi}_{ij}^{k+2} \leftarrow \tilde{\phi}_{ij}^{k+1} - \Delta t_k (\tilde{u}_{\text{ext}})_{ij}^{k+1} |\nabla\tilde{\phi}|_{ij}^{k+1}$

Perform the averaging step; for $(i,j) \in N_{nb}$:
$\phi_{ij}^{k+1} \leftarrow \frac{1}{2}\phi_{ij}^k + \frac{1}{2}\tilde{\phi}_{ij}^{k+2}$

return ϕ^{k+1} and $\vec{y}^{\,k+1}$

Algorithm 5 Efficient Level Set Method

Require: $F = N, A = \emptyset, B = \emptyset$

Build the narrowband A and B

for $k \leftarrow 0$ to $m-1$ **do**

  Evolve ϕ^k to ϕ^{k+1} using a TVD RK3 step

  if $\exists\,(i,j) \in B, \exists\,(i_n,j_n) \in \text{Neigh}(i,j) : \text{sign}(\phi_{ij}^{k+1}) \neq \text{sign}(\phi_{i_n j_n}^{k+1})$ **then**

    Reinitialize ϕ^{k+1} to a signed distance function on the entire grid

    Rebuild the narrowband A and B

  else

    Reinitialize ϕ^{k+1} to a signed distance on the narrowband ▷ This step is optional

  end if

end for

Chapter 3

Results

In this chapter, we present the methodology we used to verify the techniques introduced in the previous section and the results of applying the scheme to solving our model of interest.

3.1 Verification

To ensure the techniques discussed were implemented correctly, we used a variety of test cases for which analytical or qualitative behaviour is known. We will start with our verification of the FMM and later on, the verification of the efficient level set method as a whole.

3.1.1 The Fast Marching Method

As discussed in Section 2.2.3, we are solving (2.15); that is:

$$\begin{cases} |\nabla \phi_{\text{temp}}| = 1 & \text{on } R, \\ \phi_{\text{temp}}(\vec{x}) = 0 & \text{for } \vec{x} \in M(t), \end{cases}$$

while simultaneously constructing a scalar field u_{ext} that satisfies (2.14):

$$\begin{cases} \nabla \phi_{\text{temp}} \cdot \nabla u_{\text{ext}} = 0 & \text{on } R, \\ u_{\text{ext}} = u_n & \text{on } M(t). \end{cases}$$

We started by validating the later equation (2.15). In particular, for a circle of radius $r_c \in \mathbb{R}^+$, it is known that the signed distance function is given by:

$$\phi_{\text{circle}}(\vec{x}) = |\vec{x}| - r_c. \tag{3.1}$$

Therefore, we run Algorithm 1 on ϕ_{circle} and compute the mean, standard deviation and max of the distribution $|\phi_{\text{circle}} - \phi_{\text{temp}}|$. We use a discretization of $\Delta h = 0.05$ of the grid $R = [-8, 8] \times [-8, 8]$. The results for the first order FMM can be seen in Table 3.1.

r_c	mean	max	standard deviation
1	0.0176	0.0402	0.0101
5	0.00435	0.0587	0.00459
7	0.00483	0.0641	0.00643

Table 3.1: Metrics of $|\phi_{\text{circle}} - \phi_{\text{temp}}|$ after running the FMM on ϕ_{circ}

The error distribution is certainly in line with what one would expect and in fact, the maximum error is below or barely exceeding the discretization of $\Delta h = 0.05$. To validate whether the adapted FMM correctly determines u_{ext} according to (2.14), we considered an analytical test case and qualitative test case. In particular, for the analytical test case, we considered ϕ_{circle} as the signed distance function with the interface velocity u_n given as discussed in (2.3). Moreover, we set the position of the VSC to $\vec{y} = (0, 0)$. To determine an analytical solution to (2.14), we consider a polar coordinate system (r, θ) in which:

$$\nabla = \frac{\partial}{\partial r}\vec{e}_r + \frac{1}{r}\frac{\partial}{\partial \theta}\vec{e}_\theta,$$

such that $\phi_{\text{circle}}(r, \theta) = r - r_c$ and:

$$\frac{\nabla \phi_{\text{circle}}}{|\nabla \phi_{\text{circle}}|} = \vec{e}_r.$$

As a result, since the curvature H on the circle is $-\frac{1}{r_c}$, the interface normal velocity u_n reduces to:

$$u_n(\theta) = \frac{P\, r_c\, \vec{e}_r \cdot \vec{e}_r}{2\pi r_c^2} \frac{1}{\frac{1}{r_c}} = \frac{P}{2\pi}.$$

Clearly, the extension $u^a_{\text{ext}} = \frac{P}{2\pi}$ solves (2.14). So, we run the FMM on ϕ_{circle} with the interface normal velocity u_n above and compute the mean, max and standard deviation of the distribution $|u^a_{\text{ext}} - u_{\text{ext}}|$ on our entire grid. The results for the first order FMM using the same discretization as before can be seen in Table 3.2.

r_c	P	mean	max	standard deviation
1	1	$4.83 * 10^{-5}$	$9.94 * 10^{-5}$	$2.69 * 10^{-5}$
1	5	$2.41 * 10^{-4}$	$4.97 * 10^{-4}$	$1.35 * 10^{-4}$
5	1	$2.18 * 10^{-6}$	$3.98 * 10^{-6}$	$1.07 * 10^{-6}$
5	5	$1.09 * 10^{-5}$	$1.99 * 10^{-5}$	$5.37 * 10^{-6}$

Table 3.2: Metrics of $|u^a_{\text{ext}} - u_{\text{ext}}|$ after running the adapted FMM on ϕ_{circ}

Once again, it is safe to say that the algorithm produces a u_{ext} in line with our expectations. In fact, since the differences used for u_n are second order, the method, at least in this particular case, produces a u_{ext} that is second order accurate. Now, this analytical test case has a very simple analytical extension u^a_{ext} and so, we should also validate whether the method will produce reasonable u_{ext} in the cases it is non constant. To this end, we will consider a fixed VSC that is off-center from the starting circle. In [16], van Raaij showed that, under a starting fixed off-center VSC, the curve M should asymptotically approach a circle for which the VSC is the center. So, we simulated a fixed VSC at $\vec{y} = (0.6, 0)$ with ϕ_{circle}, $r_c = 1.0$ as our initial condition. We kept the grid identitical to our previous simulations. The results are shown in Figure 3.1. Qualitatively,

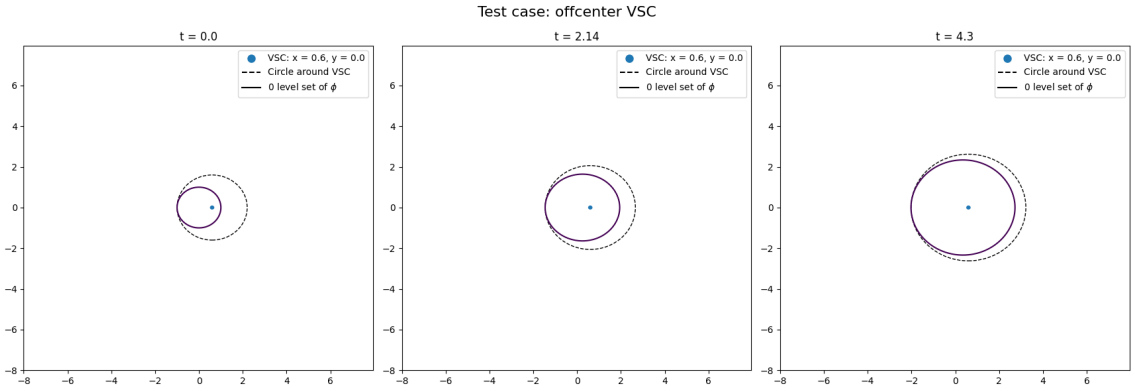


Figure 3.1: The evolution of the curve M in the case of a fixed offcenter VSC

we can indeed observe the predicted behaviour and this strongly suggests that the FMM velocity extension is working as it should in a non-uniform case.

3.1.2 The Efficient Level Set Method

After ensuring the FMM was working, we opted to validate the efficient level set method as a whole. To this end, we considered two types of flows on ϕ_{circle} for which the analytical solution is known. These were curvature flow and the u_n as defined in (2.3) with a fixed VSC at $\vec{y} = (0, 0)$. We chose to include curvature flow, as in the level set literature it is a very common flow used to showcase the methods. For curvature flow, it is well known that the circle will remain a circle as it evolves with a radius given by:

$$r_{\text{mean}}(t) = \sqrt{r_c^2 - 2t}.$$

Similarly, it can be shown that the simple ballistic VSC flow with a fixed VSC as described leads to an evolving circle with radius:

$$r_{\text{vsc}}(t) = r_c + \frac{P}{2\pi}t.$$

Since during the running of our algorithm the level set function remains a signed distance function, we opted to validate the flows by computing the error distribution between points on the discrete grid neighbouring the interface and their expected values. We ran both flows on a grid $R = [-8, 8] \times [-8, 8]$ discretized with $\Delta h = 0.05$, a narrowband of width $12\Delta h$, a starting radius of $r_c = 5$ and a production rate $P = 1.0$. In addition, we ran both with and without narrowband reinitialization after every time step[1]. The results can be seen in Table 3.3.

Clearly, Algorithm 5 was successful in simulating both flows. More noticeably, applying reinitialization on the narrowband after every time step greatly helped the overall accuracy of the method and therefore, we opted to employ this technique for all later simulations.

[1]This is the optional step in Algorithm 5

flow \ t	0.3			0.6			0.9		
	mean	max	sd	mean	max	sd	mean	max	sd
curv.	$5.03 * 10^{-4}$	$2.97 * 10^{-3}$	$6.42 * 10^{-4}$	$2.98 * 10^{-3}$	$1.85 * 10^{-2}$	$3.64 * 10^{-3}$	$7.78 * 10^{-3}$	$1.90 * 10^{-2}$	$6.09 * 10^{-3}$
curv. w. r.	$8.03 * 10^{-5}$	$1.09 * 10^{-4}$	$1.23 * 10^{-5}$	$1.13 * 10^{-4}$	$2.5 * 10^{-4}$	$1.01 * 10^{-4}$	$4.43 * 10^{-4}$	$9.97 * 10^{-4}$	$3.42 * 10^{-4}$
VSC	$4.89 * 10^{-4}$	$3.12 * 10^{-3}$	$8.38 * 10^{-4}$	$3.05 * 10^{-3}$	$1.78 * 10^{-2}$	$3.55 * 10^{-3}$	$7.92 * 10^{-3}$	$1.85 * 10^{-2}$	$2.51 * 10^{-3}$
VSC w. r.	$7.89 * 10^{-5}$	$1.21 * 10^{-4}$	$1.45 * 10^{-5}$	$1.06 * 10^{-4}$	$5.43 * 10^{-4}$	$1.15 * 10^{-4}$	$4.56 * 10^{-4}$	$1.03 * 10^{-3}$	$3.38 * 10^{-4}$

Table 3.3: Metrics of simulating curvature and fixed VSC flow

3.2 Simulation of the tethered ballistic VSC model

For the simulation, we used ϕ_{circle} as an initial condition with a radius of $r_c = 1$, $\vec{y}_0 = (0.6, 0)$ as an initial condition for the VSC and the tethering point $\vec{x}_T = (1, 0)$. Moreover, we ran it on a grid $R = [-8, 8] \times [-8, 8]$ discretized with $\Delta h = 0.05$, a narrowband of width $12\Delta h$ and a production rate $P = 2.0$. The result can be seen in Figure 3.2.

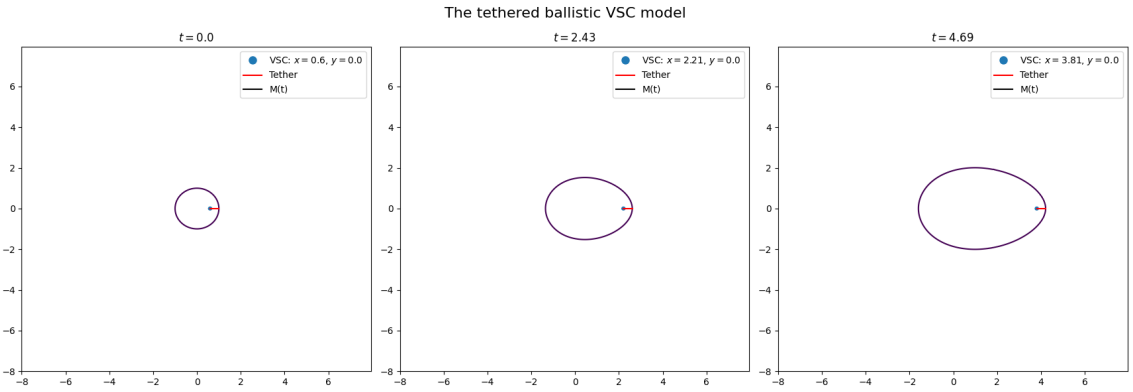


Figure 3.2: The evolution of the curve M in the tethered ballistic VSC model

We do indeed observe an elongation of the curve M that also maintains its symmetry. In that sense, the result is in line with what is qualitatively expected of tip growth in a two-dimensional setting. However, we were not able to simulate much longer than this due to encountering stability issues. In particular, as the curvature at a particular point on M approached 0, the method became unstable in those areas (see Figure 3.3). This is an inherent limitation of the two-dimensional model.

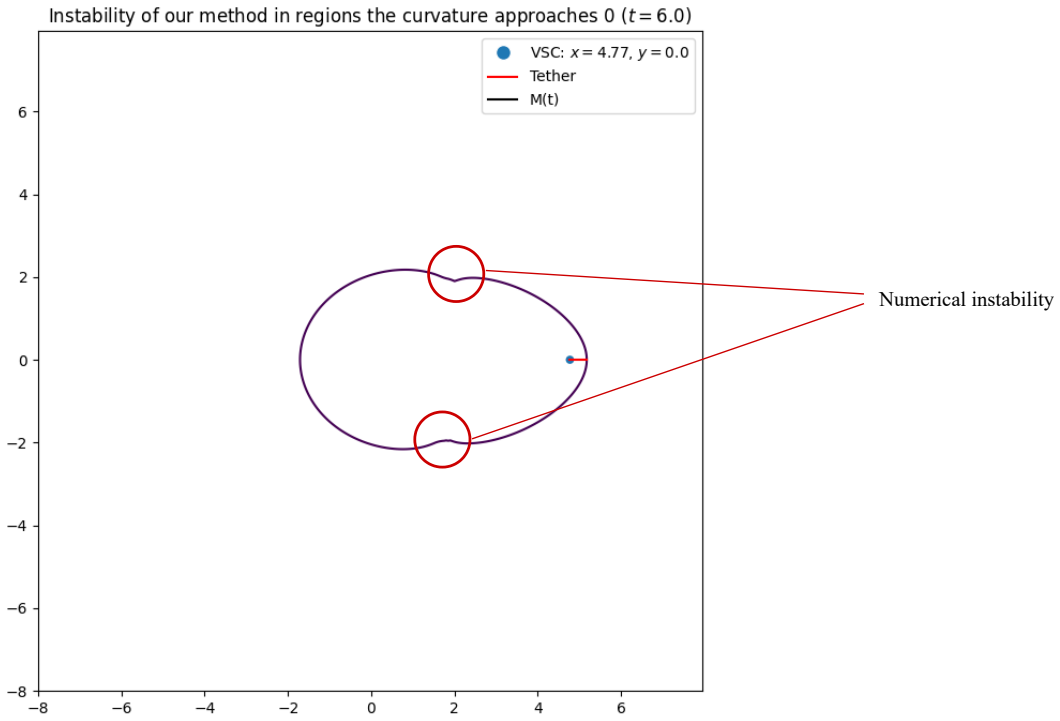


Figure 3.3: A continuation of our previous simulation until numerical instability is encountered

Chapter 4

Conclusion, Discussion & Recommendations

Conclusion & Discussion

In this report, we examined an adapted ballistic VSC model of hyphal growth. In particular, we discussed the standard assumptions; that is the idealized hypha, the Spitzenkörper and orthogonal growth along with providing the novel tethering assumption. We then combined these assumptions into the tethered ballistic VSC model. We proceeded to discuss level set methods to simulate the tethered ballistic VSC model. In doing so, we discussed standard numerical discretizations, velocity extension using the fast marching method (FMM) and the advanced level set methods narrowbanding and reinitialization. These were then all combined into a single efficient level set method. Finally, we touched upon the verification of our methods along with their implementation and ultimately, provided an example evolution of the tethered ballistic VSC model.

The main interesting outcomes of this report were the implementation of the fast marching method velocity extension and the simulation of an interface flow for which the interface velocity is inversely proportional to curvature. In the research for this report, no libraries were found implementing the FMM velocity extension and thus, the open sourcing of the code base may help others use the FMM, as a velocity extension technique. In addition, the researched level set literature did not discuss flows inversely proportional to curvature and this report experimentally shows that level set methods can successfully be used to simulate such flows.

Recommendations

Regarding the modelling section, a potential point for improvement is precisely justifying the arguments made for the tether. For instance, (1.9) could be shown explicitly and similarly, local uniqueness and existence of the coupled system (1.21) and (1.22) could be shown. A more interesting extension would be treating the tethered ballistic VSC model in a non-symmetric case. In

Section 1.2.5, we setup the tethering assumption in a general setting allowing for future work to simulate or analytically treat a more general tethered ballistic VSC model.

Another natural extension to the model is its treatment in a three-dimensional setting and in fact, this could eliminate some of the instability observed during the simulations. As we pointed out in Section 3.2, in regions where the curvature of M approached 0, our numerical scheme became unstable. In VSC models in three-dimensions, the interface velocity is inversely proportional to mean curvature [15]. Therefore, certain solutions of this higher dimensional model may avoid the complications of the curvature vanishing due to the mean curvature having an additional curvature component. Moreover, if the three dimensional model includes the assumption that the cell has a rotational symmetry, then it can be simulated on a two-dimensional grid in cylindrical coordinates. The adaptations needed to achieve this in the current code base would be quite minor and so, it would present a nice an simple extension to the work done in this thesis.

Regarding the simulation code base and numerical scheme, a few improvements can be made. For one, the FMM can be improved to higher order accuracy by employing higher order Godunov schemes and higher order interpolation while initializing the method. Preliminary work has been done in implementing the second order method, but it is still necessary to use second order interpolation while initializing the FMM. Finally, although our smoothing procedure (see Section 2.3.2) has held up during the running of simulations, it would be nice to explicitly prove its correctness.

Bibliography

[1] Yangang Xing et al. "Growing and testing mycelium bricks as building insulation materials". In: *IOP Conference Series: Earth and Environmental Science* 121 (Feb. 2018), p. 022032. ISSN: 1755-1307. DOI: 10.1088/1755-1315/121/2/022032. URL: https://iopscience.iop.org/article/10.1088/1755-1315/121/2/022032.

[2] A Ekblad et al. "The production and turnover of extramatrical mycelium of ectomycorrhizal fungi in forest soils: role in carbon cycling". In: *Plant and Soil* 366 (1 2013), pp. 1–27. ISSN: 1573-5036. DOI: 10.1007/s11104-013-1630-3. URL: https://doi.org/10.1007/s11104-013-1630-3.

[3] Roger D Finlay. "Ecological aspects of mycorrhizal symbiosis: with special emphasis on the functional diversity of interactions involving the extraradical mycelium". In: *Journal of Experimental Botany* 59 (5 Mar. 2008), pp. 1115–1126. ISSN: 0022-0957. DOI: 10.1093/jxb/ern059. URL: https://doi.org/10.1093/jxb/ern059.

[4] Myron L Smith, Johann N Bruhn, and James B Anderson. "The fungus Armillaria bulbosa is among the largest and oldest living organisms". In: *Nature* 356 (6368 1992), pp. 428–431. ISSN: 1476-4687. DOI: 10.1038/356428a0. URL: https://doi.org/10.1038/356428a0.

[5] B A Ferguson et al. "Coarse-scale population structure of pathogenic Armillaria species in a mixed-conifer forest in the Blue Mountains of northeast Oregon". In: *Canadian Journal of Forest Research* 33 (4 Apr. 2003). doi: 10.1139/x03-065, pp. 612–623. ISSN: 0045-5067. DOI: 10.1139/x03-065. URL: https://doi.org/10.1139/x03-065.

[6] S Bartnicki-Garcia, F Hergert, and G Gierz. "Computer simulation of fungal morphogenesis and the mathematical basis for hyphal (tip) growth". In: *Protoplasma* 153 (1 1989), pp. 46–57. ISSN: 1615-6102. DOI: 10.1007/BF01322464. URL: https://doi.org/10.1007/BF01322464.

[7] T G de Jong. "Topological shooting, invariant manifold theory and rigorous numerics applied to an ODE for hypha tip growth". Proefschrift. Mathematics and Computer Science, Jan. 2019. ISBN: 978-90-386-4668-8.

[8] Riquelme Meritxell et al. "Fungal Morphogenesis, from the Polarized Growth of Hyphae to Complex Reproduction and Infection Structures". In: *Microbiology and Molecular Biology Reviews* 82 (2 Apr. 2018). doi: 10.1128/MMBR.00068-17, e00068–17. DOI: 10.1128/MMBR.00068-17. URL: https://doi.org/10.1128/MMBR.00068-17.

[9] Steinberg Gero. "Hyphal Growth: a Tale of Motors, Lipids, and the Spitzenkörper". In: *Eukaryotic Cell* 6 (3 Mar. 2007). doi: 10.1128/EC.00381-06, pp. 351–360. DOI: 10.1128/EC.00381-06. URL: https://doi.org/10.1128/EC.00381-06.

[10] Salomon Bartnicki-Garcia et al. “Mapping the Growth of Fungal Hyphae: Orthogonal Cell Wall Expansion during Tip Growth and the Role of Turgor”. In: *Biophysical Journal* 79 (5 2000), pp. 2382–2390. ISSN: 0006-3495. DOI: https://doi.org/10.1016/S0006-3495(00)76483-6. URL: https://www.sciencedirect.com/science/article/pii/S0006349500764836.

[11] Arthur L. Koch. “The Problem of Hyphal Growth in Streptomycetes and Fungi”. In: *Journal of Theoretical Biology* 171 (2 Nov. 1994), pp. 137–150. ISSN: 00225193. DOI: 10.1006/jtbi.1994.1219.

[12] J Wessels, J Sietsma, and Anton Sonnenberg. “Wall Synthesis and Assembly During Hyphal Morphogenesis in Schizophyllum commune”. In: *Microbiology-sgm* 129 (June 1983), pp. 1607–1616. DOI: 10.1099/00221287-129-6-1607.

[13] Otger Campàs and L Mahadevan. “Shape and Dynamics of Tip-Growing Cells”. In: *Current Biology* 19 (24 2009), pp. 2102–2107. ISSN: 0960-9822. DOI: https://doi.org/10.1016/j.cub.2009.10.075. URL: https://www.sciencedirect.com/science/article/pii/S0960982209019836.

[14] Jacques Dumais, Sharon R Long, and Sidney L Shaw. “The Mechanics of Surface Expansion Anisotropy in Medicago truncatula Root Hairs”. In: *Plant Physiology* 136 (2 Oct. 2004), pp. 3266–3275. ISSN: 0032-0889. DOI: 10.1104/pp.104.043752. URL: https://doi.org/10.1104/pp.104.043752.

[15] J. Hulshof, R. Nolet, and G. Prokert. “Existence and linear stability of solutions of the ballistic VSC model”. In: *Discrete and Continuous Dynamical Systems - Series S* 7 (1 2014), pp. 35–51. DOI: 10.3934/dcdss.2014.7.35.

[16] van Raaij M. F. A. “Modelling evolution of boundaries in growing cells with supply centres”. Eindhoven University of Technology, July 2021.

[17] James Albert Sethian. *Level Set Methods and Fast Marching Methods*. 2nd ed. Cambridge University Press, 1999. ISBN: 978-0-521-64204-0.

[18] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Vol. 153. Springer New York, 2003. ISBN: 978-1-4684-9251-4. DOI: 10.1007/b98879.

[19] David L. Chopp. “Computing Minimal Surfaces via Level Set Curvature Flow”. In: *Journal of Computational Physics* 106 (1 May 1993), pp. 77–91. ISSN: 00219991. DOI: 10.1006/jcph.1993.1092.

[20] D Adalsteinsson and J.A Sethian. “The Fast Construction of Extension Velocities in Level Set Methods”. In: *Journal of Computational Physics* 148 (1 Jan. 1999), pp. 2–22. ISSN: 00219991. DOI: 10.1006/jcph.1998.6090.

[21] J. A. Sethian. “Fast Marching Methods”. In: *SIAM Review* 41 (2 Jan. 1999), pp. 199–235. ISSN: 0036-1445. DOI: 10.1137/S0036144598347059.

[22] Jakob Andreas Bærentzen. *On the implementation of fast marching methods for 3D lattices*. 2001.

[23] Thomas H Cormen et al. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009. ISBN: 0262033844.

[24] Y.-T. Zhang and C.-W. Shu. *Chapter 5 - ENO and WENO Schemes*. Ed. by Rémi Abgrall and Chi-Wang Shu. 2016. DOI: https://doi.org/10.1016/bs.hna.2016.09.009. URL: https://www.sciencedirect.com/science/article/pii/S1570865916300187.

[25] David Adalsteinsson and James A. Sethian. “A Fast Level Set Method for Propagating Interfaces”. In: *Journal of Computational Physics* 118 (2 May 1995), pp. 269–277. ISSN: 00219991. DOI: 10.1006/jcph.1995.1098.

Appendix A

Code

The code for the simulations in this project was written in C++. The simulations were then exported as binary files to Python for plotting details and making animations of the model. All the work has been MIT licensed and can be found at https://github.com/B0bby31/BEP_Levelset.