A FAST MARCHING LEVEL SET METHOD FOR THE STEFAN PROBLEM

by

Gabriel Wood, B.S.

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Applied Mathematics
May 2015

Committee Members:

Ray Treinen, Chair

Julio Dix

Young Ju Lee

FAIR USE AND AUTHOR'S PERMISSION STATEMENT

Fair Use

Duplication Permission

ACKNOWLEDGEMENTS

Foremost, I would like to thank my beautiful wife and children for their patience and understanding, and for continuing to put up with me while I completed this thesis. Sincere thanks to my advisor Dr. Ray Treinen who started this project with me although I had absolutely no experience in this field, worked late nights and early mornings, weekends and holidays, who pushed me through impossible deadlines and broken equipment to make sure this project was a success. I would like to thank my other committee members, Dr. Julio Dix, and Dr. Young Ju Lee for all of their feedback.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure **Page**

ABSTRACT

The Stefan problem describes the change in temperature distribution with respect to time in a medium undergoing phase change. In this thesis we provide a unique combination of established numerical techniques to solve the single phase Stefan problem in two dimensions. For this purpose it is necessary to solve the heat equation and to track the location of the free boundary as it moves. We define the finite difference method for approximating the solution to partial differential equations which forms the basis for our computations, and a collection algorithms using finite difference approximations that we use to find the solution. To track the free boundary we use a level set method, combined with a fast marching method to determine the velocity with which the boundary will move according to the Stefan condition. The heat equation is solved with a second order accurate implicit approach.

CHAPTER 1

BACKGROUND

The classical Stefan problem describes a temperature distribution in a heterogeneous medium which is undergoing a phase change, such as ice melting in water. The problem is named for J. Stefan, a physicist who worked extensively on this problem and published a series of papers on the subject in 1889, see Vuik [12] for historical background.

Stefan's original formulation of the problem was to consider a quantity of seawater which is cooled down to its freezing temperature. Assuming that no fluid is moving, suppose that at a certain time the temperature of the adjacent air decreases to α degrees below the freezing temperature of the seawater. Thereafter the temperature of the air does not change in time. Ice formation begins at the interface between air and seawater. The resulting ice layer grows as a function of time. Stefan found that the thickness of the ice layer h is proportional to the square root of the elapsed time.

A recent work on this problem uses adaptive multi grid techniques [6]. For this Thesis, our formulation of the problem, based loosely on the formulation described by Chen, Merriman, Osher, and Smereka in [2], is as follows. Consider a square domain, D, of a pure material. At each time t, the material at any grid-point is either in liquid or solid phase. Let $T(x,t)$ represent the temperature of the material. We denote the region where the material is in the liquid phase by Ω and the region where the material is solid by Ω^c. The interface between the solid and liquid pases, i.e., the boundary of Ω, is denoted by Γ.

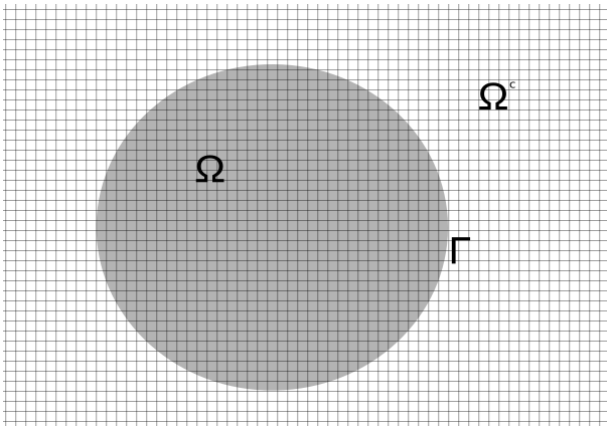

Figure 1.1: Domain

Let V represent the normal velocity at the front Γ. The governing equations for our formulation of the problem are

$$c_s \frac{\partial T}{\partial t} = \nabla \cdot (k_s \nabla T), \quad x \in \Omega^c, \tag{1.1}$$

$$c_l \frac{\partial T}{\partial t} = \nabla \cdot (k_l \nabla T), \quad x \in \Omega, \tag{1.2}$$

where c_l and c_s denote the volumetric heat capacities and k_s and k_l are the thermal diffusivities of the material in Ω and Ω^c, respectively. On Γ, the jump condition

$$L\vec{V} = -\left[k_l \frac{\partial T_\Omega}{\partial \vec{N}} - k_s \frac{\partial T_{\Omega^c}}{\partial \vec{N}}\right] \tag{1.3}$$

holds, where L denotes the latent heat of solidification. The jump is taken from liquid to solid, and the vector $\vec{N}$ is the outward normal vector at the front. In the liquid region $\frac{\partial T_\Omega}{\partial \vec{N}}$ denotes the normal derivative of T and $\frac{\partial T_{\Omega^c}}{\partial \vec{n}}$ the corresponding normal derivative of T in the solid region. Equation (1.3) is commonly referred to as the Stefan condition. For simplification scale the coordinate axes such that the constants scale to 1, and assume that the temperature in the solid region, Ω^c, will be uniformly equal to $0^\circ\mathrm{C}$. With these assumptions, equation (3) simplifies to

$$\vec{V} = -\frac{\partial T_\Omega}{\partial \vec{N}} \tag{1.4}$$

This velocity V will tell us how the boundary of the liquid region will move as the surrounding ice is melted.

CHAPTER 2

FINITE DIFFERENCE METHODS

The finite difference method is a technique for approximating the solution to a differential equation defined on a discrete grid. The finite difference approximations for the derivatives are one of the oldest numerical methods to solve differential equations and were known by Euler in 1768 when he published *Institutionum calculi integralis*. Stability and convergence criteria were established by Courant, Friedrichs, and Lewy in 1928, and by 1947 many of the formulas were already standardized as stated by Fox in [5]. The exploration of finite difference techniques in numerical applications expanded in the 1950's, and numerous papers were published on the subject during this time period.

For the sake of simplicity, consider the one-dimensional case. The main concept behind any finite difference scheme is related to the definition of the derivative of a smooth function u at a point $x \in \mathbb{R}$:

$$u'(x) = \lim_{h \to 0} \frac{u(x+h) - u(x)}{h}$$

As h tends to zero (without vanishing), the quotient on the right-hand side provides a good approximation of the derivative. We consider the approximation to be good when the error committed in this approximation tends toward zero when h tends toward zero. If the function is sufficiently smooth in the neighborhood of x, it is possible to quantify this error using a Taylor series expansion. Assuming that the function is well behaved, we can create a Taylor series expansion:

$$f(x_0 + h) = f(x_0) + \frac{f'(x_0)}{1!}h + \frac{f^{(2)}(x_0)}{2!}h^2 + ... + \frac{f^{(n)}(x_0)}{n!}h^n + R_{(n)}(x)$$

Where $R_{(n)}$ is a remainder term. The remainder term, as well as the higher order terms are small, which allows us to approximate the the first derivative as:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}. \tag{2.1}$$

With the inclusion of more neighboring grid points, higher order derivatives can be estimated in a similar fashion.

Rate of Convergence

The **order** of a finite difference method is a way of quantifying the convergence of the numerical approximation to the solution of a partial differential equation to the true solution. A finite difference stencil is first order if the error is $\mathcal{O}(\Delta x)$, second order if the error is $\mathcal{O}(\Delta x)^2$, nth order if the error is $\mathcal{O}(\Delta x)^n$ where Δx is the grid spacing in the x direction.

Spatial Derivatives

There are many different difference formulas that may be used to approximate the derivative of a function at a particular point. Depending on the numerical data, or the nature of the underlying function, we can choose the stencil that best approximates the derivative at any given point. Common stencils for first order derivatives are the forward and backwards differences, ($\mathcal{O}(\Delta x)$), and central difference ($\mathcal{O}(\Delta x)^2$):

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \qquad \text{forward difference} \tag{2.2}$$

$$f'(x) \approx \frac{f(x) - f(x-h)}{h} \qquad \text{backward difference} \tag{2.3}$$

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \qquad \text{central difference} \tag{2.4}$$

The same is true for second order or higher derivatives, though they require more points to be used to estimate the derivatives.

$$f''(x) \approx \frac{f(x-h) - 2f(x) + f(x+h)}{h^2} \qquad \text{centered difference} \tag{2.5}$$

Finite difference weights on an arbitrarily spaced grid

All of the proceeding equations assume that the function is defined on a uniform grid, this need not be the case. The following algorithm by Fornberg [3] computes the required weights for finite difference formulas on uniform, or non-uniform grids, for more details see Fornberg [4].

Given data values u_i at the locations $x_i, i = 0, 1, ..., n$, the Lagrange interpolation polynomial based on the first $j+1$ function values $u_i = u(x_i), i = 0, 1, ..., j$ becomes

$$p_j(x) = \sum_{i=0}^{j} L_{i,j}(x) u_i, j = 0, 1, ..., n,$$

where

$$L_{i,j}(x) = \frac{(x-x_0)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_j)}{(x_i-x_0)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_j)} \tag{2.6}$$

Assuming for simplicity that we want the approximations to be accurate at $x = 0$, we get

$$\left.\frac{d^k u(x)}{dx^k}\right|_{x=0} \approx \left.\frac{d^k p_j(x)}{dx^k}\right|_{x=0} \cdot u_i = \sum_{i=0}^{j} c_{i,j}^k \cdot u_i.$$

Where the second subscript for $c_{i,j}^k$ denotes the stencil width; we abbreviate $c_{i,n}^k$ as c_i^k. By Taylor's formula

$$L_{i,j}(x) = \sum_{k=0}^{j} \left.\frac{d^k L_{i,j}(x)}{dx^k}\right|_{x=0} \cdot \frac{x^k}{k!} = \sum_{k=0}^{j} c_{i,j}^k \frac{x^k}{k!}, \tag{2.7}$$

i.e., the weights $c_{i,j}^k$ can be read off from the Taylor coefficients of $L_{i,j}(x)$. Equation (2.6) implies the recursion relations

$$L_{i,j}(x) = \frac{(x-x_j)}{(x_i-x_j)} L_{i,j-1}(x)$$

and

$$L_{j,j}(x) = \left\{ \frac{\Pi_{\nu=0}^{j-2}(x_{j-1} - x_\nu)}{\Pi_{\nu=0}^{j-1}(x_j - x_\nu)} \right\} (x - x_{j-1}) L_{j-1,j-1}(x)$$

Substitution of the Taylor series (2.7) into these leads to the following recursion relations for it's coefficients:

$$c_{i,j}^k = \frac{1}{x_j - x_i}(x_j c_{i,j-1}^k - k c_{i,j-1}^{k-1}), \tag{2.8}$$

$$c_{j,j}^k = \left\{ \frac{\Pi_{\nu=0}^{j-2}(x_{j-1} - x_\nu)}{\Pi_{\nu=0}^{j-1}(x_j - x_\nu)} \right\} (k c_{j-1,j-1}^{k-1} - x_{j-1} c_{j-1,j-1}^k). \tag{2.9}$$

Starting from the trivial $c_{0,0}^0 = 1$, assuming any undefined weights to be zero, all the required weights $c_{i,j}^k$ follow recursively from (2.8) and (2.9). The ratios of the products in (2.9) are also evaluated recursively. In order to interpolate to find a the value of a partial derivative at a point in between grid points, this algorithm can be used to calculate weights for the needed order of accuracy.

Time Derivatives

Many approaches can be taken to find the time derivatives, and the best method to use will depend on the nature of the problem. The simplest technique is an **Explicit** method in which the the value at time $n + 1$ depends explicitly on the value at time n. The advantage of this method is simplicity, and computational speed. However, the use of an explicit method imposes limitations on the size of the time step that can be used, and so we are limited by the CFL condition, see Moler [7] for example, to the upper bound

$$\Delta t \leq \frac{1}{2}\Delta x^2$$

in one dimension, and

$$\Delta t \leq \frac{1}{4}\Delta x^2$$

in two dimensions. As this shows, when the grid size is decreased, the size of the time step shrinks rapidly. To get around this problem, an **implicit** method can be employed which

does not have such a time step restriction. As an illustration, if we consider the one dimensional heat equation:

$$\frac{\partial T}{\partial t} = k\frac{\partial^2 T}{\partial x^2} \tag{2.10}$$

where T represents the temperature distribution. An implicit discretization of this equation is:

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = k\frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{\Delta x^2} \tag{2.11}$$

T_i^n is the temperature distribution at the point i and at time n.

Simplifying and letting $k = 1$

$$T_i^n = (1 + 2\sigma)T_i^{n+1} - \sigma T_{i+1}^{n+1} - \sigma T_{i-1}^{n+1} \tag{2.12}$$

where $\sigma = \frac{\Delta t}{\Delta x^2}$. This gives us the current time step in terms of the following time step, and a linear system of equations must be solved in order to find the value of T_i^n. The need to solve a system of equations causes this approach to be more computationally demanding and as a result much slower than the explicit method, per time step. This increase in computational complexity is compensated for by the fact that there is no time step restriction in the implicit method for stability, so we are able to take much larger steps. However If the time step is too large it may result in an inaccurate solution so some care must be taken to maintain consistency. This method is second order accurate in space, and first order accurate in time.

The **Crank-Nicolson** method uses a combination of implicit and explicit approaches and is second order accurate in both space and time.

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

This scheme is the most accurate at small timesteps, and the implicit method is best for larger values of Δt.

CHAPTER 3

SIGNED DISTANCE FUNCTIONS

A distance function $d(\vec{x})$ can be defined

$$d(\vec{x}) = \min(|\vec{x} - \vec{x_i}|), \quad \forall \vec{x_i} \in \partial\Omega \tag{3.1}$$

Then $d(\vec{x}) = 0$ when x is on the boundary, see Osher and Fedkiw [9] for more details. If we consider a point $x \in \Omega$ which is not on the boundary and let $x_c \in \partial\Omega$ be the nearest boundary point to x. Now, imagine a line segment from x to x_c, for any point on this line segment x_c will be the closest boundary point. Using the distance function $d(\vec{x})$ to define a signed distance function

$$\phi(\vec{x}) = \begin{cases} d(\vec{x}) & \vec{x} \in \Omega \\ -d(\vec{x}) & \vec{x} \in \Omega^c \end{cases},$$

$\phi(\vec{x})$ will indicate a direction to the boundary as well as a distance, as we can see in Figure 3.1. By initializing the signed distance function such that the zero level set of $\phi(\vec{x})$ corresponds to the boundary of Ω, we can easily determine the distance to the boundary for each point $\vec{x}$, and which region of the domain it falls into, by evaluating $\phi(\vec{x})$.

The signed distance function has several useful properties that simplify calculations and enable us to represent more complicated domains simply. Let Ω be the union of pairwise disjoint closed regions $\Omega_1, \Omega_2, ..., \Omega_n$. Then
$\phi = \min(\phi_1(\vec{x}), \phi_2(\vec{x}), ..., \phi_n(\vec{x}))$ is the signed distance function representing the union of the interior regions Ω as in Figure 3.2. Where ϕ_1 represents the signed distance to $\partial\Omega_1$, ϕ_2 is the signed distance to $\partial\Omega_2$, and so forth. Similarly, if Ω is the intersection of multiple closed regions, $\phi = \max(\phi_1(\vec{x}), \phi_2(\vec{x}), ..., \phi_n(\vec{x}))$ is the signed distance function representing the interior regions of Ω. The complement of the set $\phi(\vec{x})$ is a signed distance

function, where $\phi^c(\vec{x}) = -\phi(\vec{x})$. The region $\phi_1(\vec{x}) \setminus \phi_2(\vec{x})$ has a signed distance function defined by $\phi(\vec{x}) = \max(\phi_1(\vec{x}), -\phi_2(\vec{x}))$.

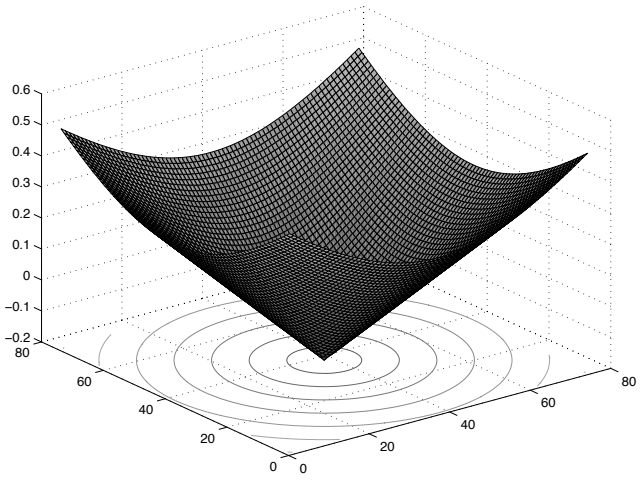

Figure 3.1: Signed distance function

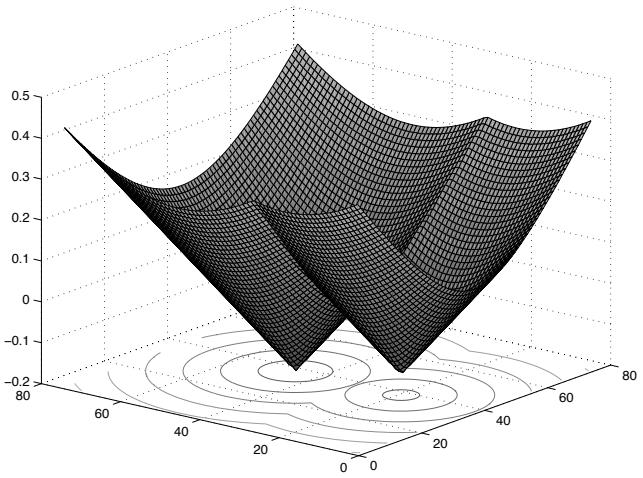

Figure 3.2: Signed distance function for disjoint warm regions

CHAPTER 4

THE HEAT EQUATION

The heat equation, also sometimes called the diffusion equation, describes the variation in heat in a given region over a period of time.

$$u_t = \gamma^2 \Delta u \qquad \text{(Heat Equation)}$$

where γ is a constant, and $\Delta u = u_{xx} + u_{yy} + u_{zz} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$

Derivation

Professor Blank of Kansas State University showed us this elegant derivation of the heat equation. We consider a three dimensional liquid region, Ω, completely contained inside a block of ice. Let $u(x, y, z, t)$ be the temperature distribution in Ω at time t and let $E(t)$ be the total amount of heat contained in Ω at time t. Then,

$$E_\Omega(t) = \int_\Omega cu \, \mathrm{d}V$$

where c is the specific heat. The change in total heat with respect to time is then given by

$$E'_\Omega(t) = \int_\Omega cu_t \, \mathrm{d}V. \tag{4.1}$$

We know by Fourier's law, see Mortimer [8] for example, that the heat flux density through the boundary of Ω is

$$\vec{q} = k \nabla u,$$

where k is the thermal conductivity of water. Since all heat must escape through the boundary of Ω, and no work is done on the system, the change in total heat is

$$E'_\Omega(t) = \int_{\partial\Omega} k \nabla u \cdot \vec{n} \, \mathrm{d}s,$$

where $\partial\Omega$ is the boundary of the liquid region. By the divergence theorem we have

$$E'_\Omega(t) = \int_{\partial\Omega} k\nabla u \cdot \vec{n}\,\mathrm{d}s = \int_\Omega k\Delta u\,\mathrm{d}V \tag{4.2}$$

By (4.1) and (4.2),

$$\begin{aligned}
\int_\Omega cu_t\,\mathrm{d}V &= \int_\Omega k\Delta u\,\mathrm{d}V \\
\int_\Omega (cu_t - k\Delta u)\,\mathrm{d}V &= 0 \\
cu_t - k\Delta u &= 0 \\
cu_t &= k\Delta u \\
u_t &= \gamma^2\Delta u
\end{aligned}$$

where $\gamma^2 = k/c$.

Numerical Solution

Using an implicit method for the time derivative and second order accurate stencils for the spacial derivatives, we can build an accurate heat equation solver that is numerically stable for larger timesteps. Given data that is not initially smooth, the heat equation will smooth the function instantaneously and the solution will remain stable despite any initial discontinuities, as can be seen in Figures 4.1 and 4.2. The solution evolves quickly, and will reach a steady state in a short period of time. We note that the steady state is the solution to Laplace's equation $\Delta u = 0$ with the same boundary data. Figures 4.3 and 4.4 were generated using our explicit heat equation solver where 500 timesteps completes in .068 seconds.

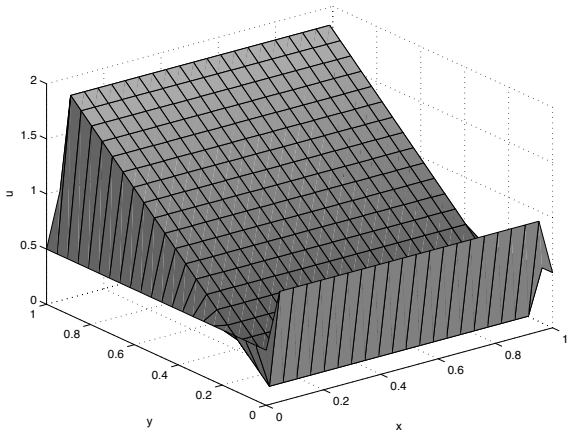

Figure 4.1: Heat distribution time $t = 0$

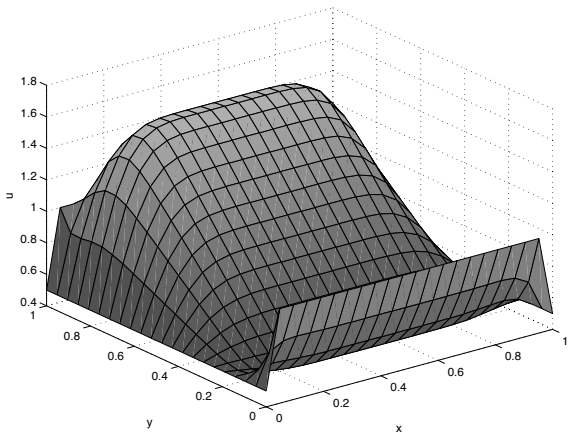

Figure 4.2: Heat distribution time $t = 10$

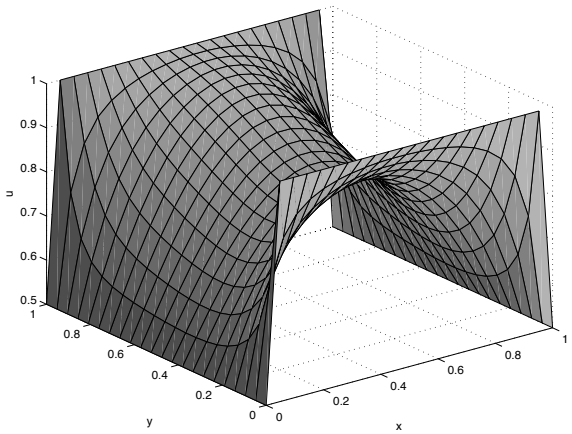

Figure 4.3: Heat distribution time $t = 100$

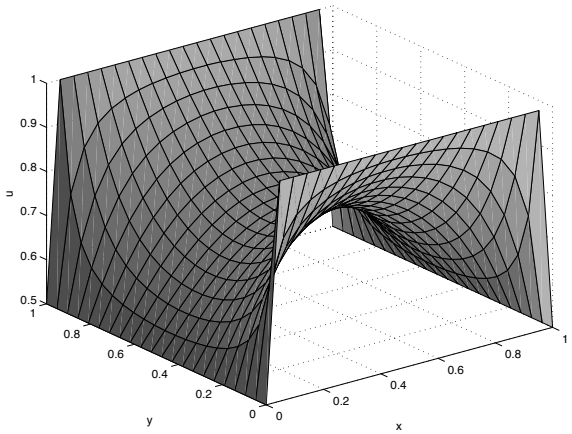

Figure 4.4: Heat distribution time $t = 500$

CHAPTER 5

THE LEVEL SET METHOD

The Level Set Method is a technique developed by Osher and Sethian in [10] to track evolving interfaces on a fixed cartesian grid. Given a domain D, and an interface Γ in R^n bounding an open region Ω, we wish to analyze and compute its subsequent motion under a velocity field, $\vec{V}$. This velocity could be due to an external force, or based on some internal physical properties as in the Stefan problem. We define a smooth function $\phi(x,t)$, called the level set function, with the interface Γ defined as the set where $\phi(x,t) = 0$, $x = x(x_1, ..., x_n) \in R^n$. The level set function ϕ is a signed distance function with the following properties

$$\begin{aligned} \phi(x,t) &> 0 \quad \text{for } x \in \Omega \\ \phi(x,t) &< 0 \quad \text{for } x \notin \overline{\Omega} \\ \phi(x,t) &= 0 \quad \text{for } x \in \partial\Omega. \end{aligned}$$

We use the function ϕ to represent the interface and also to evolve the interface. We evolve the interface in time by solving the level set equation, also called the convection or advection equation, introduced by Osher and Sethian for interface evolution. The level set equation is given by

$$\phi_t + \vec{V} \cdot \nabla\phi = 0, \qquad \text{(Level Set Equation)}$$

where ϕ_t is the partial derivative of ϕ with respect to time. This equation defines the motion of the interface Γ, where $\phi(\vec{x}) = 0$. For all later time the free boundary can be captured by locating the set $\Gamma(t)$.

We use ϕ to define the outward unit normal vector

$$\vec{N} = \frac{\nabla\phi}{|\nabla\phi|}, \tag{5.1}$$

so we see that the normal vector $\vec{N}$ points in the direction of $\nabla\phi$, which implies that for any tangential vector $\vec{T}$, $\vec{T} \cdot \nabla\phi = 0$. Letting $\vec{V} = V_n\vec{N} + V_t\vec{T}$ the level set equation is

$$\phi_t + (V_n\vec{N} + V_t\vec{T}) \cdot \nabla\phi = 0,$$

which is equivalent to

$$\phi_t + V_n\vec{N} \cdot \nabla\phi = 0.$$

Also,

$$V_n\vec{N} \cdot \nabla\phi = V_n\frac{\nabla\phi}{|\nabla\phi|} \cdot \nabla\phi = V_n\frac{|\nabla\phi|^2}{|\nabla\phi|} = V_n|\nabla\phi|,$$

so the level set equation becomes

$$\phi_t + V_n|\nabla\phi| = 0.$$

As the velocity $\vec{V}$ is only valid on the zero level set of ϕ we create a speed function $\vec{F}_{ext}$ that is a continuous extension of $\vec{V}$ off of Γ onto all of D. There will be more discussion on finding the extension velocity chapter 7. The motion is analyzed by advecting the ϕ values with the extension velocity $\vec{F}_{ext}$. Using the extension velocity, our modified level set equation becomes

$$\phi_t + \vec{F}_{ext}|\nabla\phi| = 0. \tag{5.2}$$

We compute the velocity at each grid point as a scalar speed value which is the magnitude of a velocity vector in the direction outward normal to Γ. We then solve the Level Set Equation, which in effect moves all of the level sets with the normal velocity given by $\vec{F}_{ext}$. In this way we determine the location of Γ at the next time step.

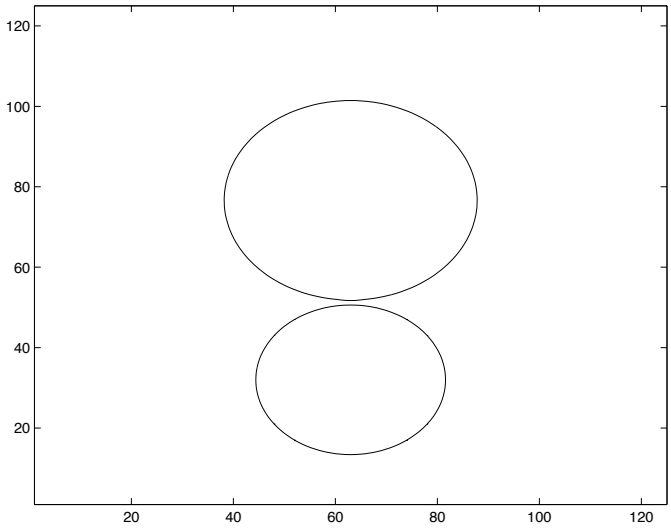

Figure 5.1: Zero level set at time $t = 0$

Using the level set approach, we create an algorithm that deals with more complicated configurations of the domain automatically and does not require any additional machinery. In instances where Ω is the union of several pairwise disjoint regions, this method will allow these regions to merge (or not) as dictated by the physical conditions of the problem. An example of this is a domain consisting of multiple warm liquid regions which are initially surrounded by ice, merging into a single closed region as the ice between them is melted. In Figures 5.1 and 5.2 we see the merging of two such regions.

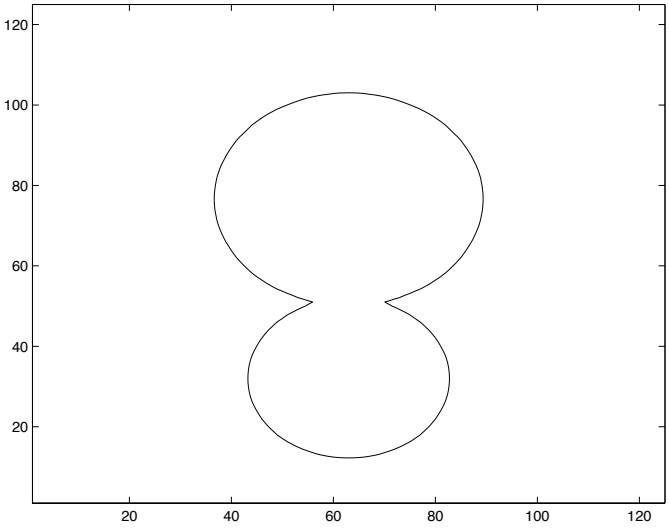

Figure 5.2: Zero level set at time $t = 15$

CHAPTER 6

HJ WENO

In order to discuss the HJ WENO method it is necessary to first go over a couple of techniques on which HJ WENO is based, and we follow the explanation by Osher in [9]. The first idea is upwinding, or upwind differencing, which allows us to choose the stencil to use in the finite difference approximations based on the direction that the relevant information is traveling. The second is the HJ ENO method, which will choose the most accurate interpolation of ϕ to use when choosing a stencil for the finite difference approximations.

Upwinding

Once we have a signed distance function, $\phi(\vec{x})$, and a velocity defined at each point in the domain, we can evolve ϕ by F to solve for the location of the boundary at the next time step. A first order accurate method for this is the forward Euler method given by:

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} + \vec{F}^n \cdot \nabla \phi^n = 0 \tag{6.1}$$

If we expand the second term, for the two dimensional case we get

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} + u^n \phi_x^n + v^n \phi_y^n = 0 \tag{6.2}$$

Where u^n and v^n are the components of $\vec{F}$ in the x and y directions, respectively, at time n. The sign of $\vec{n}$ indicates the direction of motion of $\phi(\vec{x})$ at time n. If u_i^n, u evaluated at the point $x = i$ at time n is greater than zero, then the values of ϕ are moving from left to right, so we need to look to the left of x_i to determine what value of ϕ will land on x_i at

the end of the time step. Similarly if $u_i < 0$ we need to look to the right of x_i to determine which value of ϕ will be used in updating x_i for the next time step. This information enables us to choose the best stencil to use to approximate the derivative. The points that need to be used to approximate the derivative will be those points in the direction that the information is coming from. If u_i is positive, we will need to choose a finite difference stencil that incorporates the information to the left of x_i, D_i^{-x} will give the most relevant information in this case as it uses the point to the left of x_i to calculate the derivative. The technique of choosing the approximation to be used in the derivative based on the sign of the velocity function at the point in question is called upwind differencing, or upwinding. This will bias the finite difference stencil in the direction that the most relevant information is coming from.

HJ ENO

The HJ ENO method is a way of extending the first order accurate upwinding scheme to higher spacial order by finding better approximations to ϕ_x^- and ϕ_x^+. This is accomplished by using the smoothest possible interpolation of ϕ and then differentiating to get ϕ_x. Newton's divided difference interpolation is used to interpolate ϕ, the 0th divided differences of ϕ at each grid node i located at x_i are defined by:

$$D_i^0\phi = \phi_i$$

The first divided differences of ϕ defined midway between grid nodes as

$$D_{i+\frac{1}{2}}^1\phi = \frac{D_{i+1}^0\phi - D_i^0\phi}{\Delta x}$$

The second and third divided difference are defined as

$$D_i^2\phi = \frac{D_{i+\frac{1}{2}}^1\phi - D_{i-\frac{1}{2}}^1\phi}{2\Delta x}$$

and

$\mathrm{D}^3_{i+\frac{1}{2}}\phi = \frac{D^2_{i+1}\phi - D^2_i\phi}{3\Delta x}$

where the third divided difference is defined midway between the grid nodes. These divided differences are used to create a polynomial of the form

$$\phi(x) = Q_0(x) + Q_1(x) + Q_2(x) + Q_3(x)$$

This polynomial can be differentiated in order to find the differences, ϕ_x^- and ϕ_x^+ to be used for upwinding.

$$\phi_x(x_i) = Q_1'(x_i) + Q_2'(x_i) + Q_3'(x_i);$$

To find ϕ_x^-, start with $k = i - 1$, and to find ϕ_x^+ we start with $k = i$, then define

$$Q_1(x) = (D^1_{k+\frac{1}{2}}\phi)(x - x_i)$$

then

$$Q_1'(x_i) = D^1_{k+\frac{1}{2}}\phi$$

So for ϕ_x^-

$$Q_1'(x_i) = D^1_{k+\frac{1}{2}}\phi = D^1_{i-1+\frac{1}{2}}\phi = D^1_{i-\frac{1}{2}}\phi = D^-\phi$$

and similarly for ϕ_x^+

$$Q_1'(x_i) = D^1_{k+\frac{1}{2}}\phi = D^1_{i+\frac{1}{2}}\phi = D^+\phi$$

Then, the first-oder accurate polynomial interpolation is first-order upwinding. Adding the $Q_2'(x_i)$ will get us to second order accuracy, and adding $Q_3'(x_i)$ term will lead to third order accuracy. When adding the second, and if necessary the third term, we have an option to choose to include either the point to the left or to the right. We will choose the

term based on the direction in which the data is varying more slowly, in order to avoid interpolating near discontinuities or large gradients which can cause error in the interpolating function which will then lead to errors in the approximation of the derivatives.

The result is a third-order accurate scheme that uses a subset of $\phi_{i-3}, \phi_{i-2}, \phi_{i-1}, \phi_i, \phi_{i+1}, \phi_{i+2}$ that depends on how the stencil is chosen. If we define the following $v_1 = D^-\phi_{i-2}, v_2 = D^-\phi_{i-1}, v_3 = D^-\phi_i, v_4 = D^-\phi_{i+1}, v_5 = D^-\phi_{i+2}$, then

$$\phi_x^1 = \frac{v_1}{3} - \frac{7v_2}{6} + \frac{11v_3}{6}, \tag{6.3}$$

$$\phi_x^2 = -\frac{v_2}{6} + \frac{5v_3}{6} + \frac{v_4}{3}, \tag{6.4}$$

$$\phi_x^3 = \frac{v_3}{3} + \frac{5v_4}{6} - \frac{v_5}{6} \tag{6.5}$$

are the three potential HJ ENO approximations to ϕ_x^-. The one of these approximations with the least error will be chosen by choosing the smoothest possible polynomial interpolation of ϕ.

HJ WENO

With the HJ ENO method, only one of the three candidate stencils (6.3), (6.4), or (6.5) is chosen. It turns out that in smooth regions where the function is well behaved, this is unnecessary. Much higher accuracy can be gained by using a weighted combination of the three ENO approximations, up to fifth order in smooth regions. The HJ WENO approximation of $(\phi_x^-)_i$ is given by:

$$\phi_x = w_1\phi_x^1 + w_2\phi_x^2 + w_3\phi_x^3 \tag{6.6}$$

where $0 \leq w_k \leq 1$ are the weights and $w_1 + w_2 + w_3 = 1$. It has been found that in smooth regions, the weights $w_1 = .1$, $w_2 = .6$, and $w_3 = .3$ give the optimal

approximations. In smooth regions, these weights will lead to inaccurate results. It is necessary then to base the weights on some estimate of the smoothness of the stencils in (6.3), (6.4), and (6.5). If we set

$$S_1 = \frac{13}{12}(v_1 - 2v_2 + v_3)^2 + \frac{1}{4}(v_1 - 4v_2 + 3v_3)^2, \tag{6.7}$$

$$S_2 = \frac{13}{12}(v_2 - 2v_3 + v4)^2 + \frac{1}{4}(v_2 - v_4)^2, \tag{6.8}$$

and

$$S_3 = \frac{13}{12}(v_3 - 2v_4 + v_5)^2 + \frac{1}{4}(3v_3 - 4v_4 + v_5)^2. \tag{6.9}$$

Then use the smoothness estimates and define

$$\alpha_1 = \frac{.1}{(S_1 + \epsilon)^2}, \tag{6.10}$$

$$\alpha_2 = \frac{.6}{(S_2 + \epsilon)^2}, \tag{6.11}$$

and

$$\alpha_3 = \frac{.3}{(S_3 + \epsilon)^2}, \tag{6.12}$$

with $\epsilon = 10^{-6}$ max$(v_1^2, v_2^2, v_3^2, v_4^2, v_5^2)$ + 10^{-99}, where the 10^{-99} term prevents division by zero in the α calculations. Then we set the weights as follows:

$$w_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2 + \alpha_3}, \tag{6.13}$$

$$w_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2 + \alpha_3}, \tag{6.14}$$

and

$$w_3 = \frac{\alpha_3}{\alpha_1 + \alpha_2 + \alpha_3}. \tag{6.15}$$

If the data are sufficiently smooth, these will give approximately the optimal weights $w_1 \approx .1$, $w_2 \approx .6$, and $w_3 \approx .3$. If one of the stencils incorporates a region where the data are not smooth, the corresponding value for α will be small when compared to the other

stencils, and the stencil will be given minimal weight. There can still be problems if all three stencils include data that is not smooth, but in this case the HJ ENO method would have trouble as well. If these problematic areas are localized, the method will repair itself and only have temporary, localized error as a result.

CHAPTER 7

THE FAST MARCHING METHOD

The fast marching method is a technique introduced by Sethian [11] to solve certain boundary value problems by propagating information out from a boundary at which values are known.

Finding Signed Distances

We build a signed distance function $\phi(\vec{x})$ with two properties:

1. The value of ϕ at a point $\vec{x}$ represents the signed distance from that point to the boundary.
2. The function must satisfy $|\nabla\phi(\vec{x})| = 1$.

In order to solve the equation in property 2, which is a version of the Eikonal equation, we use an upwinding approximation from Godunov:

$$[\max(D_{ij}^{-x}\phi, -D_{ij}^{+x}\phi, 0)^2 + \max(D_{ij}^{-y}\phi, -Dij^{+y}\phi, 0)^2]^{\frac{1}{2}} = 1 \tag{7.1}$$

where D_{ij}^{+x} is the forward difference in the x direction at the point (i,j), D_{ij}^{-x} is the backward difference in the x direction at the point (i,j), similarly for the y direction.

$$\begin{aligned} D^{-x}\phi &= \frac{\phi_{i,j} - \phi_{i-1,j}}{h} \\ -D^{+x}\phi &= \frac{\phi_{i,j} - \phi_{i+1,j}}{h} \\ D^{-y}\phi &= \frac{\phi_{i,j} - \phi_{i,j-1}}{h} \\ -D^{+y}\phi &= \frac{\phi_{i,j} - \phi_{i,j+1}}{h} \end{aligned}$$

This upwinding approach will propagate information from the smallest values to the largest values, so we use a systematic method to find the solution starting with the smallest values of ϕ, those which are closest to the boundary.

The first step is to locate the points which are adjacent to the free boundary. As $\phi(\vec{x}, t = 0)$ is a signed distance function we are able to easily locate these points, as any point with a neighbor who has the opposite sign will be adjacent to the boundary. Using a one grid point thick band of points adjacent to the boundary on either side the distance to the boundary can be calculated quickly by using a contour plotter, such as Matlab's contour function, to determine the zero isocontour of the signed distance function. Once the location of the boundary is known, the distance at points adjacent to the front is determined as the Euclidean distance to the closest boundary point, $d(\vec{x}) = ||x - x_c||_2$.

Now that distance to the front has been calculated for all points adjacent to the boundary we begin with the fast marching algorithm. The idea is to start with the point having the smallest distance value, and then "march" from this point calculating the values of distance as we progress away from the boundary. Since we are progressing from a closed surface, the points outside will be calculated seperately from the points inside. In other words, we march inward from the free boundary to the center, and then march outward to the edge of the computational domain. Start by partitioning the points in Ω into three sets: Accepted, Close, and Far. The initial band of calculated points is placed in the set Close, with all other points in Far, and Accepted is empty. From there we repeat this process:

1. Tag the point closest to the boundary as Trial.
2. Move the point Trial from Close into Accepted.
3. Move all neighbors of Trial that are in Far into Close.

4. Recalculate the distance to the boundary for neighboring points in Close by finding the solution to the quadratic 7.1, based only on points that are in Accepted.

5. Repeat until all points are in Accepted.

ϕ_{ext} = Accepted is now a signed distance function whose zero level set corresponds to $\phi(\vec{x}) = 0$.

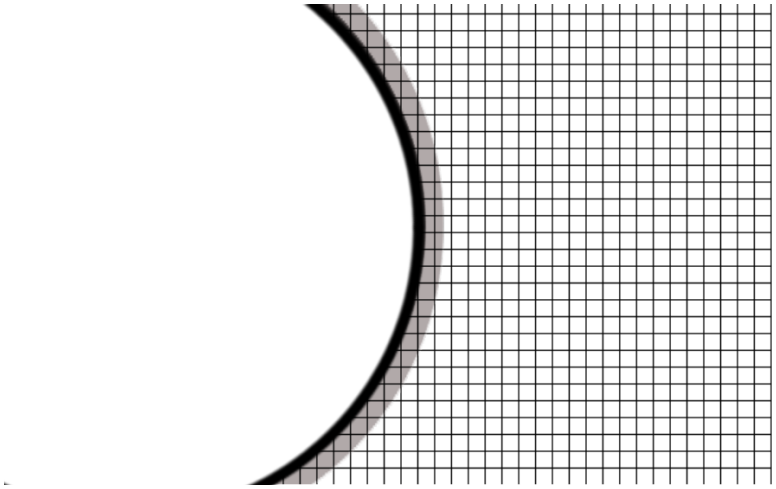

Figure 7.1: Initial band of close points

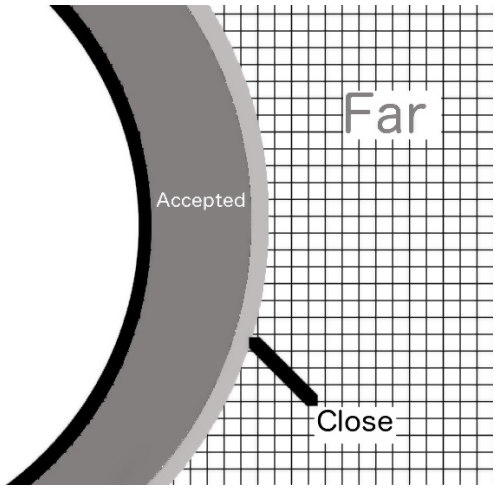

Figure 7.2: After some timesteps

In performing calculations we consider all points in Far and Close to have value infinity. Then the solution to the quadratic 7.1 is as follows:

Let $\alpha = \max(D^{-x}\phi, -D^{+x}\phi, 0)$ and $\beta = \max(D^{-y}\phi, -D^{+y}\phi, 0)$ then 7.1 is always of the form

$$\left[\left(\frac{\phi_{ij} - a}{h}\right)^2 + \left(\frac{\phi_{ij} - b}{h}\right)^2\right]^{\frac{1}{2}} = 1 \tag{7.2}$$

where a and b are defined

$$a = \begin{cases} \phi_{i-1,j} & \text{for } \alpha = D^{-x}\phi \\ \phi_{i+1,j} & \text{for } \alpha = -D^{+x}\phi \end{cases}$$

$$b = \begin{cases} \phi_{i-1,j} & \text{for } \beta = D^{-y}\phi \\ \phi_{i+1,j} & \text{for } \beta = -D^{+y}\phi \end{cases}$$

Given that only the values in accepted are finite, only one of $\phi_{i-1,j}, \phi_{i+1,j}$ and only one of $\phi_{i,j-1}, \phi_{i,j+1}$ will be finite. The corresponding difference formula will take the value $-\infty$ and will not be chosen by 7.1. The solution to 7.1 is

$$\phi_{i,j} = \frac{a+b}{h} \pm \frac{\sqrt{(a+b)^2 - 2(a^2 + b^2 - h^2)}}{2}.$$

Extension Velocity

In many problems involving a moving boundary, the velocity F will be defined at the boundary by physical conditions, or otherwise given by the model. In these cases there is some natural definition for the way that *the boundary* will evolve with time. Away from the boundary there may be no such definitions. It is necessary then, to extend the front velocity F to all of the points away from the front. The velocity at these points, denoted F_{ext}, can be constructed somewhat arbitrarily, with the only requirement being that in the limit as one approaches the zero level set F_{ext} must be equal to F:

$$\lim_{x \to a} F_{ext}(x) = F(a)$$

where a is a point on the front.

Since we are creating these values however we like, we can choose to create them in such a way as to preserve the signed distance function. If F_{ext} is constructed carefully it will avoid the stretching and bunching of the level sets that will damage the signed distance function and create the need for frequent reinitialization that can be costly in terms of computations. We construct F_{ext} such that

$$\nabla F_{ext} \cdot \nabla \phi = 0 \tag{7.3}$$

Then the level set function will remain a signed distance function for all time, as long as ϕ and F_{ext} are smooth. Since ϕ is initially a signed distance function, $|\nabla \phi(x, t = 0)| = 1$. When we move ϕ under the level set equation $\phi_t + F_{ext}|\nabla \phi| = 0$ then

$$\begin{aligned} \frac{d|\nabla \phi|^2}{dt} &= \frac{d}{dt}(\nabla \phi \cdot \nabla \phi) \\ &= 2\nabla \phi \cdot \frac{d}{dt}\nabla \phi \\ &= -2\nabla \phi \cdot \nabla F_{ext}|\nabla \phi| - 2\nabla \phi \cdot \nabla |\nabla \phi| F_{ext} \end{aligned}$$

The terms on the right are both zero by the construction of the extension velocity and the fact that $|\nabla \phi(x, t = 0)| = 1$. Hence, $|\nabla \phi|$ = 1 is a solution tho this equation and the uniqueness result for this differential equation shows that $|\nabla \phi|$ = 1 for all time.

CHAPTER 8

SOLVING THE STEFAN PROBLEM

Now that all of the pieces are in place we can solve the Stefan problem. The idea is to use the level set method to track the location of the free boundary, and then solve the heat equation inside this boundary at each time step. The algorithm is as follows:

1. Using the Fast Marching Method, determine the velocity F based on the Stefan condition and the extension velocity F_{ext} at each point in the computational domain, such that F_{ext} agrees with F at the boundary.
2. Move the level sets with this velocity, updating the position of the free boundary.
3. Solve the heat equation in Ω, determining the heat distribution at the next timestep
4. Repeat

The velocity is given on $\partial\Omega$ by the Stefan condition,

$$V = -\frac{\partial T_\Omega}{\partial \vec{n}} \tag{8.1}$$

We calculate V for each point in the initial band of close points inside the free boundary. To accomplish this, we calculate ∇T at each of these close interior gridpoints using their calculated distances to the boundary, and Fornberg's one sided difference weights to maintain second order accuracy. Once the velocity is calculated at points near the boundary, we "march" inward and outward from the boundary to every point in the computational domain calculating F_{ext} at each point used in finding the signed distance function. This method maintains the signed distance function such that it does not need to be reinitialized.

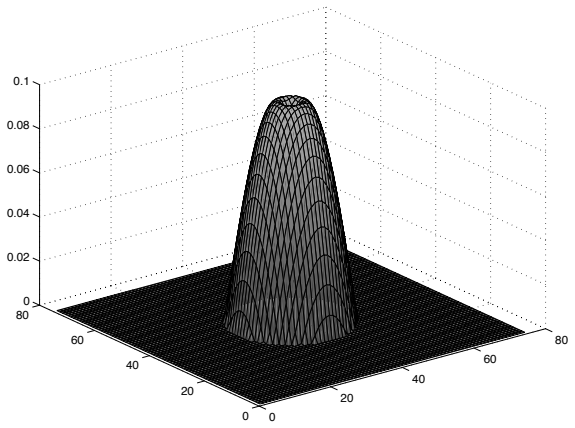

Figure 8.1: Heat distribution at time $t = 0$

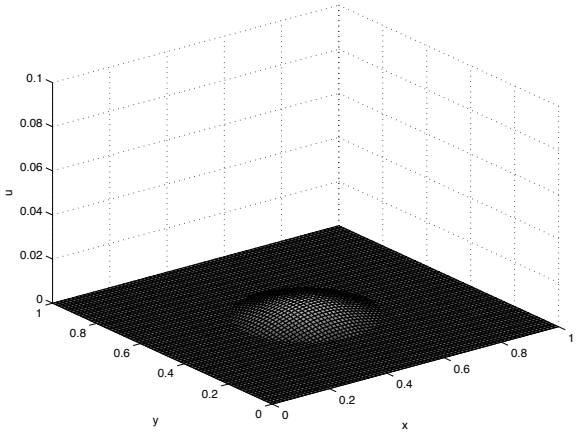

Figure 8.2: Heat distribution at time $t = 10$

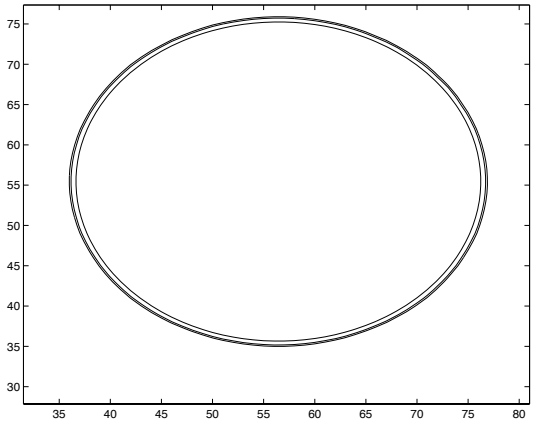

Figure 8.3: Free boundary at time $t = 0, 5, 10$

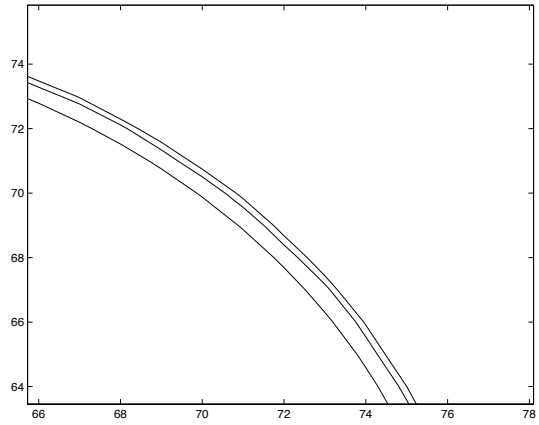

Figure 8.4: Closer view of free boundary at time $t = 0, 5, 10$

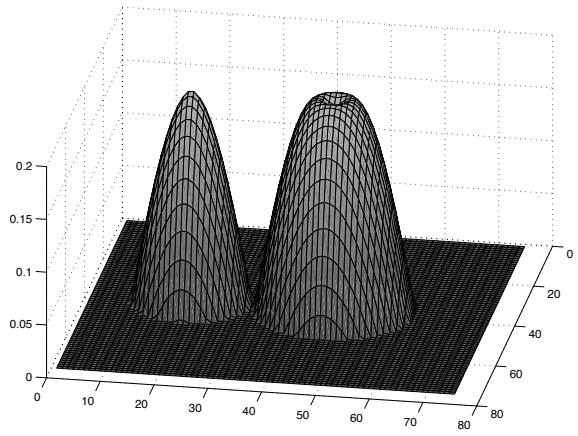

Figure 8.5: Fusing Level Sets, heat distribution $t = 0$

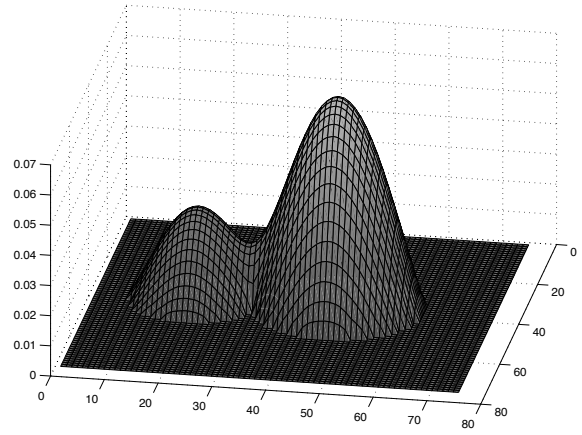

Figure 8.6: Fusing Level Sets, heat distribution $t = 10$

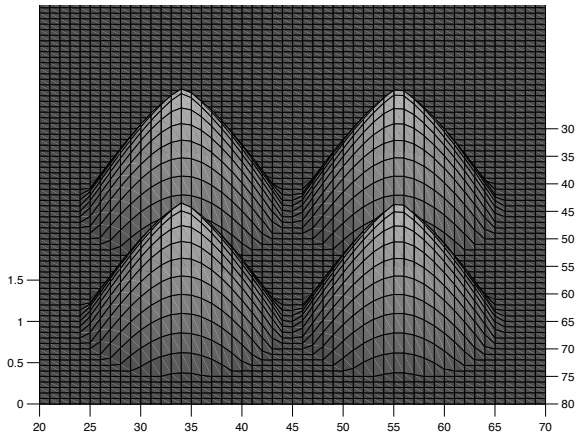

Figure 8.7: Four warm regions, heat distribution $t = 0$

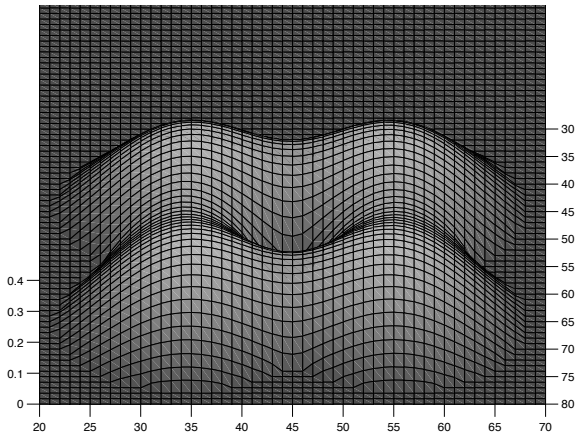

Figure 8.8: Four warm regions, heat distribution $t = 10$

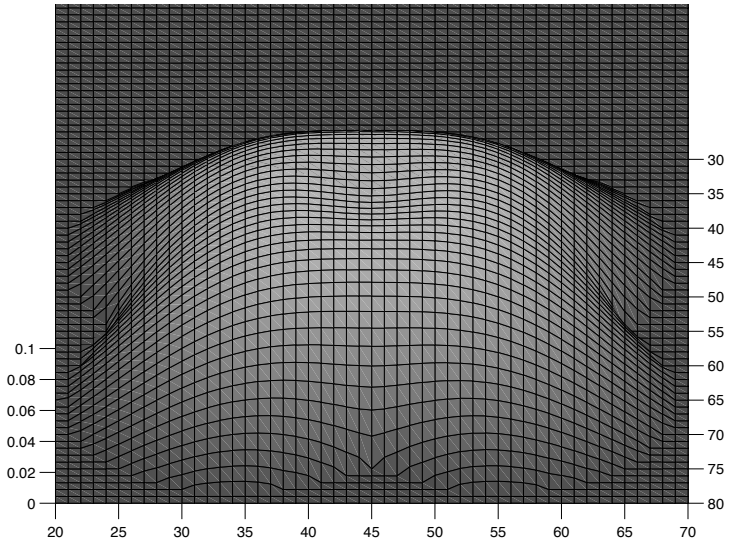

Figure 8.9: Four warm regions, heat distribution $t = 25$

BIBLIOGRAPHY

[1] D. Adalsteinsson and J. A. Sethian, *The fast construction of extension velocities in level set methods*, J. Comput. Phys. **148** (1999), no. 1, 2–22.

[2] S. Chen, B. Merriman, S. Osher, and P. Smereka, *A simple level set method for solving Stefan problems*, J. Comput. Phys. **135** (1997), no. 1, 8–29.

[3] Bengt Fornberg, *Calculation of weights in finite difference formulas*, SIAM Rev. **40** (1998), no. 3, 685–691 (electronic).

[4] _____, *A practical guide to pseudospectral methods*, Cambridge Monographs on Applied and Computational Mathematics, vol. 1, Cambridge University Press, Cambridge, 1996.

[5] L. Fox, *Some improvements in the use of relaxation methods for the solution of ordinary and partial differential equations*, Proc. Roy. Soc. London. Ser. A. **190** (1947), 31–59.

[6] Maxime Theillard, Chris H. Rycroft, and Frédéric Gibou, *A multigrid method on non-graded adaptive octree and quadtree Cartesian grids*, J. Sci. Comput. **55** (2013), no. 1, 1–15, DOI 10.1007/s10915-012-9619-2. MR3030701

[7] Cleve B. Moler, *Numerical computing with MATLAB*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2004.

[8] Robert Mortimer, *Physical Chemistry*, Elsevier Academic Press, Burlington, MA, 2008.

[9] Stanley Osher and Ronald Fedkiw, *Level set methods and dynamic implicit surfaces*, Applied Mathematical Sciences, vol. 153, Springer-Verlag, New York, 2003.

[10] Stanley Osher and James A. Sethian, *Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys. **79** (1988), no. 1, 12–49.

[11] J. A. Sethian, *A fast marching level set method for monotonically advancing fronts*, Proc. Nat. Acad. Sci. U.S.A. **93** (1996), no. 4, 1591–1595.

[12] C. Vuik, *Some historical notes on the Stefan problem*, Nieuw Arch. Wisk. (4) **11** (1993), no. 2, 157–167.