Motion by Mean Curvature: The Phase-field Method

Dante Amoroso
Worcester Polytechnic Institute

August 6, 2008

Abstract

The phase-field method is applied to the problem of motion by mean curvature. Simulations are performed doing motion by mean curvature explicitly on an initial sine wave, and analytical calculations are done for the effect of mean curvature governed evolution on an initial circle. The phase-field method is then used to solve the same situations and excellent agreement is found. The computation time of the phase-field method is then investigated, and a method involving Fourier transforms is implemented to speed up this computation. Error is introduced by this procedure, but two possible solutions are proposed. Due to time constraints, unfortunately, neither solution could be investigated. Lastly, the Fourier transform phase-field method is applied to a random initial condition and is seen to reliably produce domains which become more homogeneous with time, a result expected for any interface governed by mean curvature.

1 Introduction

Motion by mean curvature refers to studying an interface whose evolution in time is governed by Eq. 1.1, where v is the normal velocity of the interface, κ is the curvature at that point, and ν is a constant. It is used to model many phenomena involving separate non-conserved phases which evolve to reduce local surface area at an interface. Its applicability can be seen from the fact that the minimum local surface area is achieved by a perfectly straight line.

This also corresponds to zero curvature. Any small instances of curvature beyond this will go away with time as the materials seek to minimize the surface area at which they contact, thus minimizing their energy. Clearly, the more curved the surface, the faster this will occur.

$$v = \nu\kappa \tag{1.1}$$

Equation 1.1 can also be thought of from a more atomic standpoint. If an interface is perfectly flat, the atoms lying on the interface will have a certain number of bonds with like atoms and a certain number with atoms on the other side of the interface. As the straight line is at equilibrium, this configuration is clearly preferred by the atoms. Should the interface contain a bulge into one side, the atoms doing the bulging will be less bonded to like atoms than normal, while the atoms being bulged into will be more bonded than usual. Thus, both sides will move to reclaim the equilibrium number of bonds, and the interface will level out.
In this paper I will discuss several aspects of solving Eq. 1.1 in two dimensions (a one dimensional interface). In Section 2, the domain is discretized along one dimension and a straightforward finite difference method is employed to solve for the motion of the interface. In Section 3, the domain is discretized along both dimensions and an approach called the 'Phase-field method' is used. This is seen to have the advantage of being able to handle more complicated interfaces than the first case, but the disadvantage of taking much longer for each iteration to be computed. Lastly, in Section 4, the phase-field computation is done in fourier space through the use of a method involving the fourier transform of the differential equation which normally governs the phase-field evolution. This removes an inherent constraint on the time step for the straightforward phase-field method, thus allowing much larger time steps and a much shorter total computation time.

2 Explicit Treatment

In this section, we begin by changing Eq. 1.1 into a form that is easily computed. As we are dealing with a 2-D interface, we will henceforth refer to the axis along which the interface extends as the x-axis, while the axis along which the interface varies in location as the h-axis. In this way, our interface is completely described by a single function $h(x)$ at any desired point in time. Of course, Eq. 1.1 contains neither an h nor an x, and thus

must be restated before computation may begin.
As the parameter ν is a constant, we may leave it as is, and proceed to examine the κ term first. Were we dealing with a higher dimensional system, this κ would represent the mean curvature, commonly given by the sum of the principal curvatures. In the simple case of a 1-D interface, however, it is just the curvature. For ordinary functions such as our interface, this curvature is given by Eq. 2.1.

$$\kappa = \frac{\frac{d^2h}{dx^2}}{\left(1+\left(\frac{dh}{dx}\right)^2\right)^{\frac{3}{2}}} \quad \text{or} \quad \frac{\nabla^2 h}{\left(1+(\nabla h)^2\right)^{\frac{3}{2}}} \tag{2.1}$$

However, the velocity v in Eq. 1.1 is normal to the interface, requiring the relation in Eq. 2.2 to be established. From here it is straightforward to simply plug Eq. 2.2 and Eq. 2.1 into Eq. 1.1, resulting in the final Equation 2.3 describing the motion of the interface as time goes on.

$$\frac{\partial h}{\partial t} = v\sqrt{1+(\nabla h)^2} \tag{2.2}$$

$$\frac{\partial h}{\partial t} = \nu\frac{\nabla^2 h}{1+(\nabla h)^2} \tag{2.3}$$

At this stage it is important to note that Eq. 2.3 is basically a nonlinear diffusion equation. This is important as large curvatures decay towards flatter surfaces, causing the gradient term in the equation to go to zero and the nonlinearity to vanish. Thus, for small slopes we can expect Eq. 2.3 to behave just as the diffusion equation would. It is commonly known that an initial condition of a sine wave in the diffusion equation produces a perfectly exponential decay in amplitude. Thus, if we utilize a sine wave as the initial condition and track its amplitude over time, we expect a less-than-exponential drop initially, gradually approaching an exponential drop. Additionally, the decay constant should simply be $\omega^2\nu$, where the initial state is $h(x) = A\sin(\omega x)$. We will use this simple fact to examine the accuracy of numerical calculations.

2.1 Finite Difference Methods

As noted in Section 1, we solve Eq. 2.3 by discretizing along the x-axis and employing the Euler method in time and using a finite difference approximation for the spatial derivatives. The general form of the Euler method can be seen in Eq. 2.4. The derivation of finite difference formulae is slightly longer, although equally straightforward. We begin by finding several Taylor expansions, seen in Eq. 2.5.

$$f(t+\Delta t) = f(t) + f'(t)\cdot \Delta t \tag{2.4}$$

$$\begin{aligned}
f(x+2\Delta x) =& f(x) + 2(\Delta x)f'(x) + 2(\Delta x)^2 f''(x) + \frac{4(\Delta x)^3}{3}f'''(x) + \\
& \frac{2(\Delta x)^4}{3}f^{(4)}(x) + \mathcal{O}(\Delta x^5)\\
f(x+\Delta x) =& f(x) + (\Delta x)f'(x) + \frac{\Delta x^2}{2}f''(x) + \frac{\Delta x^3}{6}f'''(x) + \\
& \frac{\Delta x^4}{24}f^{(4)}(x) + \mathcal{O}(\Delta x^5)\\
f(x+(1/2)\Delta x) =& f(x) + \frac{\Delta x}{2}f'(x)\frac{\Delta x^2}{8}f''(x) + \mathcal{O}(\Delta x^3)\\
f(x-(1/2)\Delta x) =& f(x) - \frac{\Delta x}{2}f'(x)\frac{\Delta x^2}{8}f''(x) + \mathcal{O}(\Delta x^3)\\
f(x-\Delta x) =& f(x) - (\Delta x)f'(x) + \frac{\Delta x^2}{2}f''(x) - \frac{\Delta x^3}{6}f'''(x) + \\
& \frac{\Delta x^4}{24}f^{(4)}(x) + \mathcal{O}(\Delta x^5)\\
f(x-2\Delta x) =& f(x) - 2(\Delta x)f'(x) + 2(\Delta x)^2 f''(x) - \frac{4(\Delta x)^3}{3}f'''(x) + \\
& \frac{2(\Delta x)^4}{3}f^{(4)}(x) + \mathcal{O}(\Delta x^5)
\end{aligned} \tag{2.5}$$

We can now experiment with these various equations to arrive at several approximations for $f'(x)$ and $f''(x)$ to various orders of accuracy. The logical starting place is clearly the expression $f'(x) = \frac{f(x+\Delta x)-f(x)}{\Delta x}$, called the 'forward difference' approximation. A quick glance at the Taylor expansion for $f(x+\Delta x)$, however, indicates that this will only be accurate to $\mathcal{O}(\Delta x)$. The backward difference, $f'(x) = \frac{f(x)-f(x-\Delta x)}{\Delta x}$, is the same. However, using the central difference, seen in Eq. 2.6, we achieve accuracy to $\mathcal{O}(\Delta x^2)$. This type of reasoning gives us several finite difference formulae. The second derivative $f''(x)$ is given to $\mathcal{O}(\Delta x^2)$ by Eq. 2.7. The first derivative to $\mathcal{O}(\Delta x^4)$ is given by Eq. 2.8, and the second derivative to $\mathcal{O}(\Delta x^4)$ is shown in Eq. 2.9.

$$f'(x) = \frac{f\left(x + \frac{1}{2}\Delta x\right) - f\left(x - \frac{1}{2}\Delta x\right)}{\Delta x} \tag{2.6}$$

$$f''(x) = \frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{\Delta x^2} \tag{2.7}$$

$$f'(x) = \frac{-f(x+2\Delta x) + 8f(x+\Delta x) - 8f(x-\Delta x) + f(x-2\Delta x)}{12(\Delta x)} \tag{2.8}$$

$$f''(x) = \frac{-f(x+2\Delta x) + 16f(x+\Delta x) - 30f(x) + 16f(x-\Delta x) - f(x-2\Delta x)}{12(\Delta x)^2} \tag{2.9}$$

Once all these equations are derived, programming a solver for Eq. 2.3 is as simple as applying Eq. 2.4 and plugging in Eqs. 2.8 and 2.9. The x-axis is divided into a large number of subdivisions, usually $N = 1024$, and Δx becomes the natural division between these, i.e. 1. To ensure that terms like $h(x+2\Delta x)$ make sense when $x = N$, periodic boundary conditions are used. Thus, rather than referring to $h(x+2\Delta x)$ as $h(i+2)$, a vector called `ipp()` is created and defined as `ipp(i)=i+2` except for the special cases of `ipp(N-1)` and `ipp(N)`. Thus, the main loop can be executed by the relatively simple FORTRAN77 code:

```
 do i=1,N
    change(i)=dt*v*((-h(ipp(i))+16*h(ip(i))-30h(i)
+         +16*h(im(i))-h(imm(i)))/12)/(1+((-h(ipp(i))
+         +8*h(ip(i))-8*h(im(i))+h(imm(i)))/12)**2)
 enddo
 do i=1,N
    h(i) = h(i)+change(i)
 enddo
```

This code was run on a variety of sine waves, so as to compare with the expected behavior as near-linear conditions were achieved. One of these results, for the case of an initial condition of a sine wave with four periods and an amplitude of 100 ($N = 1024$) is shown in Figure 1. It is clearly seen from the graph that after the amplitude has decreased sufficiently, the nonlinear equation does in fact have comparable slope to the linear expectation. This is confirmed by fitting an exponential to the late data, also shown in the figure.

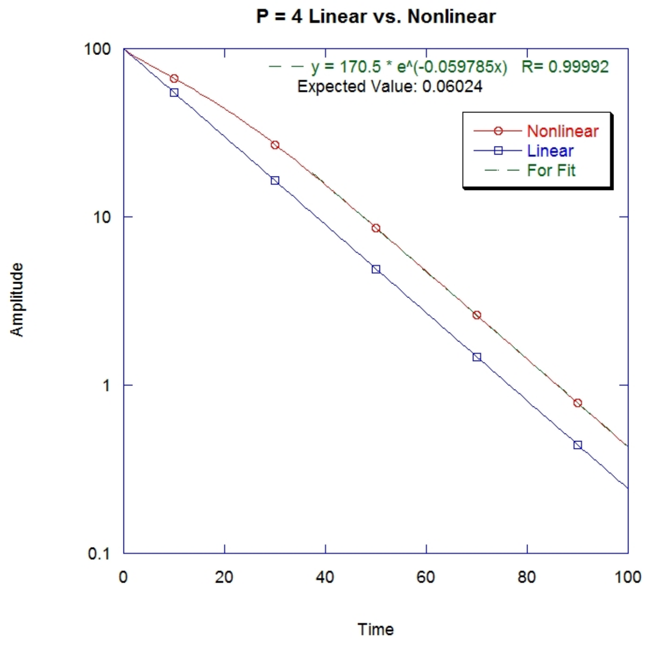

Figure 1: Comparison of results of motion by mean curvature program with expected results for linear diffusion equation. It can be plainly seen that for later times, the program results do approach an exponential decay. When these late points are fitted with an exponential curve (labeled 'For Fit' on plot) it is seen that they are in fact strongly exponential and with a decay constant very near the expected value.

3 The Phase-Field Method

The method just described has one major limitation, however. The function $h(x)$ must be a true function. That is, it can only take one value in a given column. This prohibits interfaces from taking shapes such a circles, or having multiple interfaces at once. Resolving this problem is the purpose of the phase-field method.

In the phase-field method, we begin by discretizing both the x and the h axes. For simplicity, we will now refer to the h-axis as the y-axis, as both axes assume an equal role from now on, and x-y is standard terminology. The x-axis still runs from 1 to N, while the y-axis runs from 1 to M. Typical values for N and M are $N = 1024$ and $M = 512$, although $M = 1024$ is also

sometimes used. We then introduce the phase parameter ϕ, which takes a value ranging from -1 to 1 at each point on the grid. These two extremes correspond to opposite sides of the interface, with the interface itself residing at $\phi = 0$. This results in a large bulk at either -1 or 1, with interfaces near 0 surrounded by a width of values between 0 and the extremes.
Having established *what* the phase-field is, we now need to describe how it changes to reflect the process of motion by mean curvature. We begin by establishing a free energy functional, seen in Eq. 3.1. In this, the gradient term represents the free energy cost associated with having interfaces between the two phases, while the function $f(\phi)$ is a double well potential with zeroes at -1 and 1 which serves to preference any given cell into choosing a phase rather than staying at an intermediate value. The evolution of the phase-field with this free energy is then given by Eq. 3.2.

$$\mathcal{F}(\phi) = \int dx^2 \left[\frac{1}{2}V\left(\nabla\phi\right)^2 + f(\phi)\right], \qquad f(\phi) = a\left(\phi^2 - 1\right)^2 \tag{3.1}$$

$$\frac{\partial\phi}{\partial t} = -G\frac{\delta\mathcal{F}}{\delta t} \tag{3.2}$$

How to actually compute Eq. 3.2 is not immediately apparent, though, as we still have the $\frac{\delta\mathcal{F}}{\delta t}$ on the right hand side. This is known as a functional derivative. Although the actual definition of a functional derivative can be a bit cumbersome, for functionals of the form of Eq. 3.3 the functional derivative can be expressed by Eq. 3.4. As our free energy functional from Eq. 3.1 is of that form, we can simply apply Eq. 3.4 to it and arrive at Eq. 3.5.

$$\mathcal{F}\left[\phi\left(\vec{x}\right)\right] = \int d\vec{x}\; g\left(\phi\left(\vec{x}\right), \vec{\nabla}\phi\left(\vec{x}\right)\right) \tag{3.3}$$

$$\frac{\delta\mathcal{F}}{\delta\phi\left(\vec{x}\right)} = \frac{\partial g}{\partial\phi\left(\vec{x}\right)} - \vec{\nabla}\cdot\frac{\partial g}{\partial\vec{\nabla}\phi\left(\vec{x}\right)} \tag{3.4}$$

$$\frac{\partial\phi}{\partial t} = -G\left[-V\nabla^2\phi + \frac{\partial f}{\partial\phi}\right] \tag{3.5}$$

Once this relation was obtained we utilize the same methods as Section 2, namely Euler for time and finite difference for space, with the sole difference that, as the Laplacian was now in two dimensions, we only employ the finite difference approximation which is good to $\mathcal{O}(\Delta x^2)$ (Eqs. 2.6 and 2.7). Again, a snippet of code detailing the main loop is presented below. Periodic boundary conditions are still applied, and terms such as `xp(i)` refer to the next point along the x-axis. A difference between `xp(i)` and `yp(i)` is necessitated by the possibility of non-square domains (i.e. $N \neq M$).

```
 do i=1,N
    do j=1,M
       change(i,j)=dt*-G*(-V*(p(xp(i),j)+p(i,yp(j))-4*p(i,j)+
+           p(xm(i),j)+p(i,ym(j)))+4*a*p(i,j)*p(i,j)**2-1))
    enddo
enddo
do i=1,N
   do j=1,M
      p(i,j) = p(i,j)+change(i,j)
   enddo
enddo
```

3.1 Results

One of the first things to note about Eq. 3.5 versus Eq. 2.3 is that in the phase-field method we end up with two constants (G and V), while in the original 'dh/dt' method we only had one (ν). According to Ref. [2], however, these constants are related by Eq. 3.6. In fact, in both equations the constants can be scaled out entirely. Without scaling, however, we first want to test how changing the amount of ν in G versus in V might effect the results. As the original sine wave results were obtained with $\nu = 100$, V was set to the values $\{1, 2, 4, 10, 100\}$, with the corresponding G in each case such that $G \cdot V = 100$. In all treatments of sine waves from here on, a period of four was used for the initial conditions. Figure 2 shows how the drop in amplitude changes for the different balances of constants, namely, $G = 1, V = 100$ having perfect agreement, while $G = 100, V = 1$ never changed at all.

$$\nu = G \cdot V \tag{3.6}$$

Although initially confused by the data in Figure 2, we also looked at the width of the interface for these cases. This data is shown in Figure 3. Although the data gets cluttered towards the bottom, each set of constants has a maximum and minimum width at each measured timestep, and each set shows the same trend with time. Thus, looking at the data for $G = 1, V = 100$, we notice that as time goes on, the minimum width stays constant while the maximum width slowly approaches it. However, this is

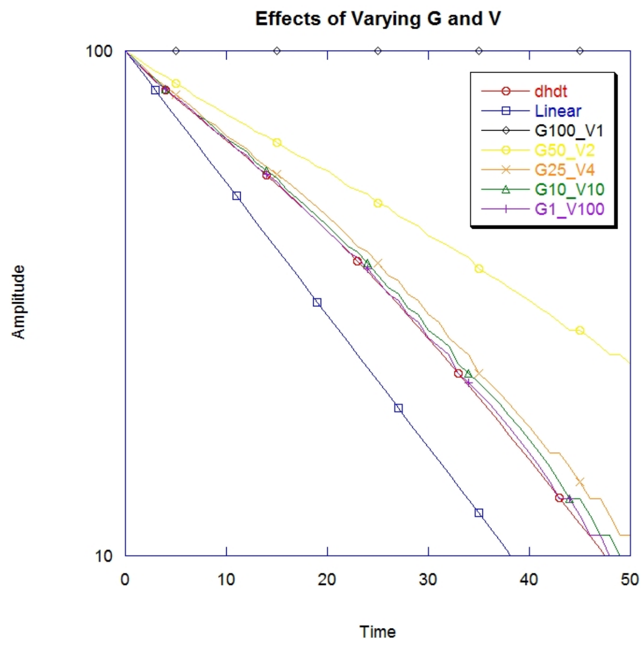

Figure 2: Comparison of different weighting between G and V for the case $G \cdot V = 100$. The 'dh/dt' results for a four period sine wave are presented for reference, along with the linear prediction. From the graph it can be seen that full weighting on V produced the most accurate results, while full weighting on G produced absolutely no motion whatsoever.

completely explained by the method of determining width in the program. The data is simply read in and scanned vertically for each column along the x-axis. When $\phi = -0.8$ (10% of the maximum range) it starts counting the number of grid sections passed until surpassing $\phi = 0.8$ (90% of the range). If the slope is high, however, this does not give a good estimate of the width, as a true width would be normal to the interface. Closer investigation confirmed that maxima were at high slope and minima were at zero slope, and correcting the maxima for this effect results in a single width for all time.
Once this is taken into account, it is seen that weighing the constants more towards G leads to thinner and thinner interfaces. With this in mind, we see that the discrepancies between amplitude drops in Figure 2 is simply due to numerical error as a result of having too thin an interface for the computer to properly simulate the effects. This leads us to confirm that Eq. 3.6 is indeed

satisfied, and simply use values $G = 1, V = 100$ in future calculations. There is one final important note to make about Figures 2 and 3. All values for phase field results are given in integers. This is because the programs look for the first cell to satisfy a certain criterion, and simply use the location of that cell from then on. This leads to slightly inaccurate graphs, a point which will be addressed later.

We now proceed to verify the accuracy of our phase field treatment in one

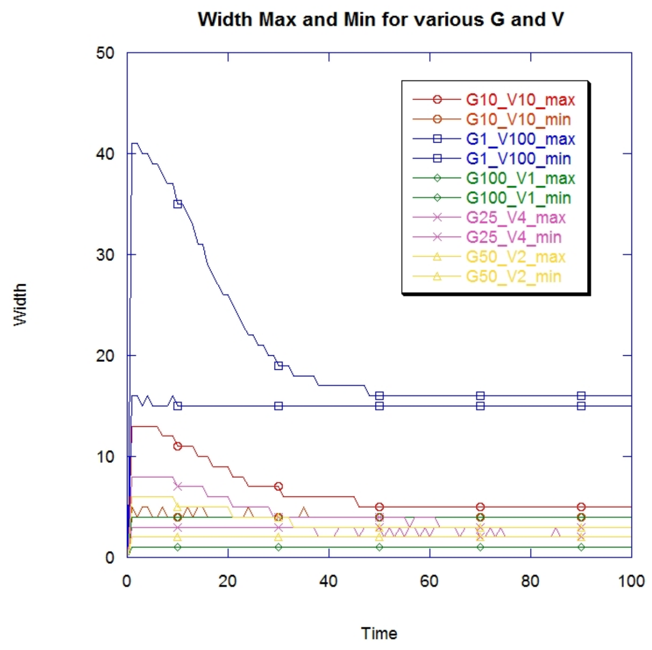

Figure 3: Plot of maximum and minimum width of interface for four period sine waves with time for various combinations of G and V. The confusing data at the bottom simply shows the same general trend visible at the top: maxima approach minima with time, while minima are constant, and higher G leads to a thinner interface.

other way: by using a circle as the initial condition. Although this situation can't be treated by our initial 'dh/dt' program, it *is* easily treated explicitly by Eq. 1.1. For a circle, curvature is simply the inverse of the radius, and the normal to the edge of a circle is always pointing along the radial axis, thus we can simply write Eq. 3.7, which has solution Eq. 3.8. Several trials with circles of initial radii $R_0 = \{200, 150, 100, 50, 10\}$ were run, and the results plotted against the expectation of Eq. 3.8. As can be seen by Figure 4, the phase field program agrees very nicely with expected results.

$$\frac{dR}{dt} = -\frac{1}{R} \tag{3.7}$$

$$R(t) = \sqrt{R_0^2 - 2t} \quad \text{or} \quad A(t) = A_0 - 2\pi t \tag{3.8}$$

Once these tests are complete, we are able to simulate what is arguably the

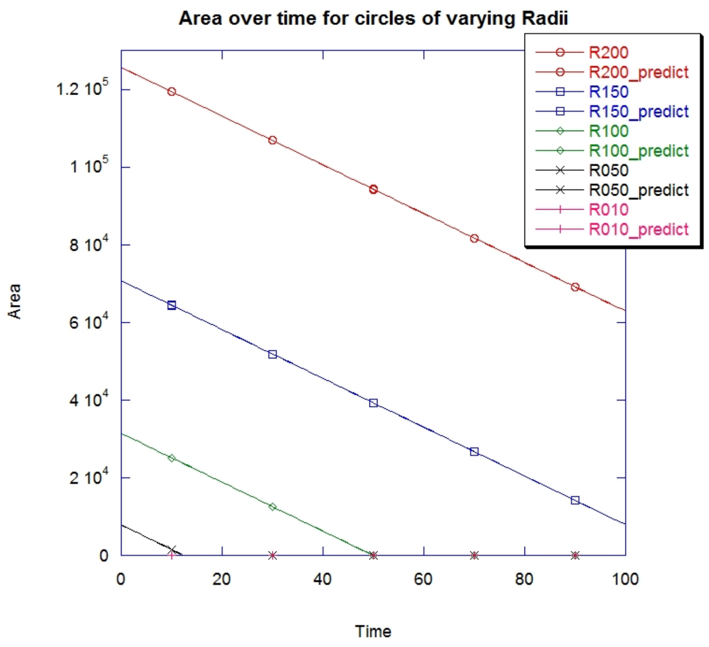

Figure 4: Plot of circle area over time for various initial radii and comparison to theory. From here it can be seen that near perfect agreement is achieved between the phase field program and the expected results of motion by mean curvature.

most complicated interface possible: a completely random initial condition. For a number of trials of varying total time movies were generated showing a random initial distribution coarsening into large domains of one phase or the other, then slowly smoothing out bumps along the interfaces. This type of calculation would be completely impossible using the straightforward 'dh/dt' method, but also highlights the single largest downfall of the phase-field method. When doing sine wave simulations in the 'dh/dt' program, a typical program runtime is on the order of 10-15 minutes. Doing the same calculation with the phase-field method takes closer to 5 hours. This is a result of the large regions of uniform phase where the computer spends most of its time computing zero change. In an effort to accelerate these sorts of computations, we now turn to Fourier transforms.

4 Fourier Transforms

The advantage of working in Fourier space rather than x-y space is entirely due to what happens when you take the Fourier transform of the Laplacian operator, namely, that it becomes simply a constant. Ordinarily, the accuracy of the finite difference approximations used constrains time steps for Laplacians according to Eq. 4.1. However, in Fourier space our equation becomes Eq. 4.2. In this formula, Φ is the Fourier transform of ϕ, thus becoming a function of k_1 and k_2 rather than x and y. In addition, the term $\{\tilde{f}'(\phi)\}_{\vec{k}}$ simply refers to the Fourier transform of the $\frac{\partial f}{\partial \phi}$ term from Eq. 3.5. Since the evolution of each point in Fourier space is independent of surrounding points, we lose the instability that a Laplacian can induce at large time steps, allowing us to have very large time steps in our computation.

$$\Delta t \lesssim (\Delta x)^2 \tag{4.1}$$

$$\frac{\partial \Phi}{\partial t} = -G\left[-V\left(-4\pi^2\left(k_1^2+k_2^2\right)\right)\Phi + \left\{\tilde{f}'(\phi)\right\}_{\vec{k}}\right] \tag{4.2}$$

The actual formula for the computation begins by re-writing Eq. 4.2 in simpler form as Eq. 4.3, where L is a linear operator (the constants in front of the first Φ) and NL is some nonlinear operator (the $\{\tilde{f}'(\phi)\}_{\vec{k}}$ term). From here we start to solve the equation using Eq. 4.4, and reach the final step by assuming the nonlinear portion of the equation is constant within the integral over the time step being used. Eq. 4.5 is rewritten in terms of all explicit constants.

$$\frac{\partial \Phi}{\partial t} = \mathrm{L}\Phi + \mathrm{NL} \tag{4.3}$$

$$\begin{aligned}\Phi(t+dt) =& e^{\mathrm{L}dt}\Phi(t) + e^{\mathrm{L}dt}\int_0^{dt} e^{-\mathrm{L}t'}\mathrm{NL}(t')dt' \\ \approx & e^{\mathrm{L}dt}\Phi(t) + \frac{e^{\mathrm{L}dt}\Phi(t)-1}{\mathrm{L}}\mathrm{NL}(t)\end{aligned} \tag{4.4}$$

$$\Phi(t+dt) = e^{-GV4\pi^2\left(k_1^2+k_2^2\right)dt}\Phi(t) + \frac{e^{-GV4\pi^2\left(k_1^2+k_2^2\right)dt}\Phi(t)-1}{V4\pi^2\left(k_1^2+k_2^2\right)}\left\{\tilde{f}'(\phi)\right\}_{\vec{k}}|_t \tag{4.5}$$

The general structure of the program is now given below. As a final note, in order to be able to use Fast Fourier Transforms (FFTs) both N and M must be a power of two.

- Initial Condition
- Evaluate $f'(\phi)$
- Take Fourier transforms of ϕ and $f'(\phi)$
- Compute next time step (Eq. 4.5)
- Do inverse Fourier transform of resulting Φ
- Repeat with this ϕ as the new initial condition
- At regular intervals, take resulting ϕ and output data

4.1 Results

We once again start with the initial condition of a four period sine wave and plot the amplitude with time. Two time steps were used in testing the new method, $dt = 0.1$ and $dt = 0.05$. These contrast to the much smaller interval being used in all previous computations of $dt = 0.001$. Although taking Fourier transforms greatly increases the amount of computation needed at each time step, the drastic reduction in the number of time steps allows our program to speed up significantly regardless. In the case of $dt = 0.05$, the program took about one hour to evaluate, while $dt = 0.1$ was about a half hour. This lets the programs retain the advantage of handling complex interfaces that the phase field method provides, but returns computation times to something much closer to those of the 'dh/dt' program.
This speed increase does come at a cost, however. Figure 5 shows the comparison of these two trials, along with the $G = 1, V = 100$ phase field trial

from before, the ‘dh/dt’ result, and the linear expectation. Additionally, for this graph, rather than taking the first grid space to satisfy a given condition as the height of the interface, the value at the point closest to zero and the points above and below this point were fit linearly to obtain a continuous measure of interface location in the phase field cases. This greatly clarifies the graph, allowing us to see the the phase-field method without fourier transforms really is exactly reproducing the ‘dh/dt’ method. It also allows us to see that there is some error present when using Fourier transforms to speed up computation. Additionally, this error is lessened by in the smaller time step, suggesting it is numerical in nature. I feel that the most probable

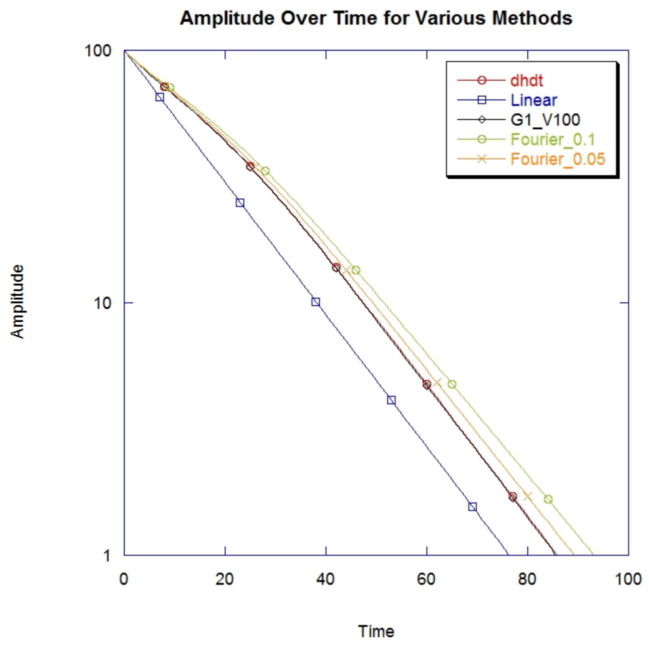

Figure 5: Final comparison of all methods employed as pertaining to amplitude decay of an initial sine wave. The heights for the phase-field methods were obtained by doing a linear fit to the point closest to zero along a given x location with the points immediately above and below it. This removes the stepping seen in previous plots of phase-field height and clearly shows the accuracy of normal phase-field versus the error of introducing Fourier transforms.

source of this error is in the approximation step within Eq. 4.4, although I was not able to investigate that in detail. In the current setup, to achieve closer accuracy it appears we will need an even smaller time step. This will very quickly negate the advantages of using Fourier transforms as a source of speed increase. One possible solution, however, lies in removing the step computing the inverse Fourier transform of Φ at the end of each time step. Currently, this is necessary because there is no other way to get the Fourier transform of the nonlinear portion without first evaluating it in x-y space. The methods proposed in Ref. [3], however, should allow this step to be bypassed, requiring an inverse Fourier transform *only* when data is desired as output. Another possible solution would be to use the schemes presented in Ref. [1] instead of Eq. 4.4, as this paper claims superior accuracy over more common methods. Unfortunately, I did not have time to investigate either route.

The final computation in this project was to take the Fourier approach to the phase-field method (despite its slight inaccuracy) and compute another random initial condition. In this way I was finally able to extend the idea of motion by mean curvature to a complicated interface, and still compute it in a very reasonable amount of time, the crowning achievement of my REU.

5 Conclusions and Possible Future

Overall, the use of the phase-field approach to problems involving motion by mean curvature was successfully implemented. With both sine waves and circles for initial conditions, computed behavior agreed essentially flawlessly with expected behavior. Additionally, we were able to model the very complicated case of a random start and see larger scale structure arise. In an attempt to speed computation, Fourier transforms proved to be very successful, despite introducing some slight error into the result. Potential workarounds for these errors do exist, however, and just could not be investigated in the time alotted.

This work could hopefully be extended in the future in a number of ways. In the immediate short-term, the methods of Orzag and/or Chen et. al. can be implemented and their effectiveness investigated. In the longer-term, now that general inroads into the implementation of phase-field has been made, the method might be applied to new situations that haven't been dealt with via phase-fields before, such as the growth of films studied by Dr. Amar's research group. Additionally, I personally hope to be able to carry some of these methods and ideas back to my University in the fall, where I will be doing a senior project on the dynamics of ferrofluid materials. These goals seem very achievable at the current time, and hopefully will be realized in the days, months, or years to come.

References

[1] L. Chen and J. Shen. Applications of semi-implicit fourier-spectral method to phase field equations. *Comp. Phys. Comm.*, 108:147–158, 1998.

[2] K. Elder, M. Grant, N. Provatas, and J. Kosterlitz. Sharp interface limits of phase-field models. *Phys. Rev. E*, 64, July 2001.

[3] D. Gottlieb and S. Orzag. *Numerical Analysis of Spectral Methods: Theory and Applications.* SIAM-CBMS, Philadelphia, PA, 1977.