Modeling Crystal Growth in a Diffusion Field Using Fully-Faceted Interfaces

Andrew R. Roosen
National Institute of Standards and Technology
and
Jean E. Taylor
Math Department, Rutgers University

Subject classifications: 65C20, 80A20.
Keywords: crystal growth, solidification, crystalline curvature, faceted interfaces.

Running head: Modeling Crystals Using Faceted Interfaces

Mailing author:
Andrew R. Roosen
National Institute of Standards and Technology
Materials A153
Gaithersburg, MD 20899

Abstract

We present a new computational model of crystal growth, in which the interface between liquid and solid is explicitly tracked, but measurement of curvature is simplified through the assumption that the crystal is a polygon having a limited number of possible normal directions. This method has several advantages. Computations involving the motion of the interface are relatively fast as compared to 'phase field' algorithms, but unlike many 'curve tracking' methods, it is easy to detect and make topological changes. The computational algorithm is described including a method for 'shattering' interface edges. The effects of variations of both physical (surface energy, mobility) and non-physical (mesh size) computational parameters have been investigated and produce results consistent with theory.

1 Introduction

The study of solidification is significant from three perspectives. Even the most superficial observer must notice the wonderfully intricate patterns formed in a snowflake. A more practical observer would realize that a study of crystallization could have a profound impact on industrial applications. Finally, a theoretician would note that a snowflake is a remarkable example of spontaneous pattern formation. These three motivations, the aesthetic, the practical, and the theoretical, all drive the study of crystallization.

Physically, a crystal is an ordered lattice of atoms, in a solid [**?**, p. 20]. The types of atoms, the particular ordering, and the extent of the ordering all have important effects on the macroscopic behavior of the crystal. In the 1930's and 40's, Ukichiro Nakaya discovered that the structure of a snowflake was determined by the temperature and the humidity of the atmosphere in which it formed [**?**]. Further, if the conditions changed during growth, the type of growth changed instantaneously [**?**, **?**]. This suggests that the growth of snow crystals should be explainable through the use of a relatively simple collection of rules. As a more practical example, when alloys solidify, small treelike (dendritic) structures form [**?**, p. 94]. The size and shape of these structures can have large effects on the strength and other properties of the alloy, and it can be observed that changing the temperature or concentration of impurities in the alloy change the way these structures form [**?**, ch. 10].

Even with a simple model of crystallization, the actual interface produced may be extremely complex. For the standard 'sharp interface' model, analytic solutions are possible in only a very few cases [**?**]. Several computational schemes have been implemented [**?**], but computational solutions have their own problems. Models of crystal growth typically involve a reduction in the total free energy of the system, the free energy consisting of a 'bulk' term that causes the system to freeze, and a 'surface' term which penalizes the creation of new interface and which is described locally by the curvature of the interface. This makes computations of the motion somewhat tricky, as the simplest direct approach requires the measurement of the second derivative of the curve. Several methods have been used to approach the problem indirectly by not actually measuring the curvature, but they tend to be computationally expensive and restricted in the functional forms of the surface energies that can be modeled. The 'crystalline curvature' method presented here, on the other hand, can be used both to model truly faceted surface energies, as well as to approximate smooth ones. It uses an interface tracking scheme that avoids most of the problems associated with other interface tracking schemes, since it does not require the measurement of second derivatives and it can quickly detect topological changes. Motion by crystalline curvature alone is discussed in [**?**], and computations with it are shown in [**?**]. It is also straightforward to investigate the effects of changes in the governing equations. In the model presented here, interface motion is heavily influenced by a temperature field. A theoretical model describing the motion of faceted curves in a diffusion field was discussed in [?, ?] under the implicit assumption that the number of edges may not increase, but this assumption is clearly non-physical in many cases, and we will describe a method to determine the appropriate 'breaking points' or 'shattering' of an edge under the influence of a temperature field.

2 Theoretical background

The fundamental driving equation of our model is that the rate of advance of the interface at any point s is given by:

$$v(s) = M(s)(-u(s) + \overline{\sigma} H_\Phi(s)) \tag{1}$$

M, the *mobility*, models the attachment kinetics of the interface, or the rate at which the interface can respond to the *driving force* $-u + \overline{\sigma} H_\Phi$ (where $\overline{\sigma}$ is a scaling factor). In this paper we make the assumption that M depends only on the orientation of the interface. The two terms of the driving force model the effects of temperature (or equivalently concentration of an impurity) and the surface energy between the solid and the liquid. The undercooling u, which we take to be dimensionless [?], satisfies a diffusion equation modified by a term for the release of latent heat at the interface:

$$\frac{\partial u}{\partial t} = \nabla^2 u + v(x,t)\delta_S \tag{2}$$

Note that as written here, the diffusion coefficient is the same in the solid and in the liquid, but this is not required computationally. $\overline{\sigma}$ is a scaling factor for the surface energy, and H_Φ is the *weighted mean curvature* of the curve, with respect to the surface energy functional $\Phi : \mathbb{R}^2 \to \mathbb{R}$ which is assumed to be convex and homogeneous of degree 1.

The weighted mean curvature may be defined in several ways [?], but the most appropriate for studying crystalline curvature is the following: the weighted mean curvature at a point p on a curve S in $\mathbb{R}^2$ is the negative of the rate of change of surface energy with volume swept out by appropriate deformations near p in the direction of normals to the surface divided by the average energy density. Exactly which types of deformations are 'appropriate' depends on the nature of Φ [?].

For any surface energy density function $\Phi : \mathbb{R}^2 \to \mathbb{R}$, we define the *Wulff shape* to be the set

$$\mathcal{W}_\Phi = \{x \in \mathbb{R}^2 : x \bullet u \leq \Phi(u)\ \forall \text{ unit vectors } u \in \mathbb{R}^2\} \tag{3}$$

If the surface energy functional Φ is such that the Wulff shape $\mathcal{W}_\Phi$ is a polygon, we say that Φ is *crystalline* or *fully faceted*. For such surface energies, the weighted mean curvature assumes a particularly simple form. Let S be a closed polygonal curve having only those normal directions found in the Wulff shape $\mathcal{W}_\Phi$, such that any two adjacent edges have normals which are adjacent in the Wulff shape (zero-length edges having the appropriate normals are assumed to be inserted otherwise). Then by admitting only variations which move an entire edge e, the weighted mean curvature can be seen to be [**?**]

$$H_\Phi(e) = -\frac{(\delta^+(e) + \delta^-(e))}{2}\Lambda(e)/l(e) \tag{4}$$

for e, where δ^+ is ± 1 depending on whether S is convex or concave at the positive (left) end of e (with respect to its orientation), and similarly for δ^-. $\Lambda(e)$ is the length of the edge in the $\mathcal{W}_\phi$ having the same exterior normal as e, and $l(e)$ is the length of e itself. (See Figure **??**). If the length $l(e)$ is zero, then $H_\Phi(e)$ is $\pm\infty$ or 0 according to the sign of $-(\delta^+ + \delta^-)$. Of course, edges with $H_\Phi(e) = \pm\infty$ would immediately move to lengthen in motion by (crystalline) curvature [**?**, **?**].

3 Computational Algorithm

Armed with an expression for weighted mean curvature for faceted interfaces, as well as an appropriate method for shattering edges under the influence of the undercooling field, it was possible to write a computer program to model the evolution of faceted curves in a diffusion field.

The computer program is written in C++, an object-oriented language. This means that computer structures can closely emulate the physical structures they are to represent. Each class defines certain pieces of information which are stored by objects of that class, as well as functions (known as *methods*) which are used to manipulate the object's data. Computations have two major parts: setting the initial conditions, and moving the crystal. Obviously, the first is executed only once, while the second is iterated until the preset termination conditions are met.

3.1 Basic structures

The problem of determining the development of the system may be divided into two parts, motion of the interface and the evolution of the temperature field. Three basic classes are used describe the interface. The 'edge' class contains information pertaining to a single edge. The 'interface' class is intended to represent a single crystal. Third, a 'Wulff' class contains information common to all crystals sharing the same surface energies. The temperature field is represented, by a member of the 'field' class.

3.1.1 Field

Objects of the field class are representations of scalar functions defined over the computational domain. Function values are stored on a rectangular grid, and values between grid points are determined through bilinear interpolation. Since the primary purpose of the class is to provide a way to approximate a temperature field, it contains a method to solve the heat equation on the field for a specified time. Methods also exist to efficiently initialize and adjust grid values. Not a part of the field class proper, but related to it, is code which calculates and stores an approximation to the characteristic function of a crystal. By subtracting characteristic functions of interfaces in successive time steps, the correct amount of latent heat may be released behind a moving interface.

3.1.2 Wulff

The 'Wulff' class contains information about the surface energy and mobility. Arrays store allowed normal directions, the energies in those directions, and the mobilities in those directions. Also stored are several frequently used values: the length of the edge in a given direction, trigonometric functions of the angles between consecutive normals. The Wulff object is intended to store information about the anisotropy of the system, storing a 'normalized' surface energy function. The actual capillary length used in the computation is generally given by an *energy multiplier* chosen by the user, and the mobilities by a *mobility multiplier*.

3.1.3 Interface

The interface object's primary purpose is to keep track of all the edges in the crystal, and to adjust their combinatorics as necessary. It also contains a pointer to a Wulff object which defines the allowed directions, energies and mobilities for edges in the crystal, and a pointer to a field object. Pointers to edges act as the heads of lists. The 'master list' is simply a list of all edges in the interface. It stores no information about the orientation of the edges. Lists stored for each set of parallel edges make it possible to easily locate topological changes in the crystal. By sorting each of these lists by the perpendicular height of the edge from the origin, the computation can determine when two edges with opposite orientations have passed each other, and hence may have produced a change in the topology of the crystal. The class defines methods to perform actions such as movement or shattering on all edges 'simultaneously'. Other methods weed out edges which have shrunk to zero length, and to adjust the combinatorics when oppositely oriented edges have passed each other. The interface object can also make reports about the size and shape of connected components, removing those which shrink to zero size.

3.1.4 Edge

In principle, each edge only requires four pieces of information to define it. A point in $\Re^2$ (the 'location' of the edge), and one of the allowed normal directions describe the line containing the edge, while pointers to the two adjacent edges delimit the actual line segment that is the edge. In practice, however, several other pieces of information are stored as well. For simplicity in keeping information in the interface current, each edge has a pointer to the interface object of which it is a part. Each edge is also a node in several linked lists, and must contain pointers to the next and previous members of each list. Finally, there are variables devoted to bookkeeping and increasing execution speed by storing frequently used values.

Several methods are defined to report and adjust characteristics of the edge. In the first category are methods to report the right and left endpoints of the edge, the length of the edge, the minimum, maximum, and mean temperature along the edge, and the weighted mean curvature of the edge. Important methods of the second type are those to move and shatter the edge (see the sections on movement and shattering below).

3.2 Initialization and termination

The dendritic growth simulations shown below begin with a single seed centered in the computational domain. The shape and size of this seed are read in from a text file prepared by hand. Typically, the starting seed is Wulff-shaped, but if any normal directions have been excluded, they are added by the program before actual computation begins. The starting seed is generally chosen just large enough to grow due to the undercooling rather than shrink due to surface energy effects, and so the initial temperature field is simply assigned a constant value. In some simulations it is desirable to start with a somewhat larger initial seed, in which case the interior of the seed is raised to a higher temperature and heat is allowed to diffuse on the temperature field so as to approximate the situation after a small period of growth. The user also chooses values for various operating parameters such as the energy and mobility multipliers mentioned in section 3.1.2, the density of the temperature grid, and the *minimum edge length* which will be discussed in 3.3.3 below. Computations are generally continued until the temperature field on the boundary of the computational domain has changed to a significant degree: in an experiment the boundary would effectively be infinitely far away, and so it should not be allowed to effect the computations.

3.3 Evolution Algorithm

The evolution of the system has two parts: motion of the interface and evolution of the temperature field. Interface motion has three fundamental steps: movement, merging, and shattering, and the field is updated in the heat release, and heat flow steps. Note that for the most part these steps are independent of one another and it doesn't matter in what order they are performed. In fact, the 'interface' steps may be applied several times between heat release and flow steps if, for example, a very small edge moving very rapidly results in a very small time step, but little heat release. It is important, of course, that each step be performed frequently enough in the computational time scale that inaccuracies do not develop, and that the heat be diffused for the same period of time as is allowed for interface motion.

3.3.1 Movement

The system of ordinary differential equations describing the movement of the interface is solved using a variation on the standard finite-difference (Euler) method. For each edge e, the velocity in the normal direction of the edge is calculated according to the formula $v(e) = M(n(e))(-\overline{u}(e) - (\delta^+ + \delta^-)\overline{\Phi}\Lambda(\vec{\nu}_i)/2l(e))$. Two forms of $\overline{u}(e)$ have been used: the low temperature along the edge, and the mean temperature along the edge. Both methods have some physical justification, but require slightly different computational schemes, particularly in the shattering step described below. The low temperature is determined by computing the temperature at many points along the edge and taking the minimum. The mean temperature can be computed exactly by taking advantage of the bilinear interpolation. It should be noted that it would not be difficult to use a different formulation of the edge velocity, so long as an appropriate shattering routine is written to match. The speed thus obtained is averaged with the speed determined during the last step in order to smooth out local errors as much as possible.

Once the speed of each edge is determined, a time step is determined. The step may not be larger than a maximal preset time step. Further, no edge may move further than a preset distance (generally 1/4 of a temperature grid square). Finally, and most importantly, no edge with non-zero curvature can shrink to zero length. This is because the curvature term in the speed of the edge would go to infinity as the edge shrank to zero length. Note that the size of the time step thus changes at nearly every iteration. Once the time step Δt has been chosen, the location $\mathbf{x}$ of each edge is moved a distance $v(e)\Delta t$ in the normal direction of the edge.

3.3.2 Merging

The merging step performs four types of adjustments to the combinatorial structure of the interface, i.e. how the edges abut each other. First, the whole interface is checked for 'small' connected components. If a connected sequence of edges contains an area smaller than one grid cell and the area enclosed is shrinking, those connected edges are all removed. Secondly, any 'flipped' edges are removed. An edge has 'flipped' if its normal direction points into the solid (see Figure **??**). Next, adjacent parallel edges are merged into a single edge by removing one. Generally the only way two adjacent edges become parallel is by the removal of flipped edge between them. Finally, any topological changes in the crystal are detected though the use of linked lists of parallel edges ordered by perpendicular distance from the origin. If two edges change their order, then it is possible a topological change must be made. Once the possibility of a topological change is detected, it is easy to actually make the change. Figure **??** shows an example of a very heavily undercooled computation on an underresolved temperature grid which has produced a great deal of dense side branch activity. It can be seen that many topological changes have occurred; see the region near coordinates (2.5,4.5), for example. Also note the small area of liquid completely surrounded by solid near coordinates (6,3). The fact that topological changes may be quickly detected in this way is one of the strengths of this method: there are over 15000 edges in Figure **??**, and checking such a complex figure for crossings every time step would be a formidable task for most computational methods.

3.3.3 Shattering

The shattering step is key to the simulation of dendritic growth. In this step, zero-length edges are inserted so as to allow parts of an initial edge to move faster than other parts. To date, two different algorithms have been used for determining when and where these zero-length edges be inserted.

The first algorithm is general purpose, and would be reasonable to use with any choice of the movement algorithm mentioned in 3.3.1, at the cost of being more inexact and time consuming. It makes the assumption that if enough edges are inserted, the best shattering will determine itself, relying on 'trial and error'. Specifically, zero-length edges are inserted midway between every adjacent pair of temperature critical points along the edge, oriented so that the break will expand if the colder side should move faster. The whole interface is then moved and any of the new edges which have flipped are removed. Finally, all edges are returned to their original position so that the deleted edges will have no effect on the bona-fide movement step.

The more 'exact' algorithm is tailored for using the mean temperature along an edge to determine its velocity and makes use of the analysis in [**?**]. In practice, only two types of temperature distributions are found along an edge: monotonic from one end to the other, or a single local maximum or minimum in the interior of the edge, and the algorithm assumes this is always the case. The edge is then broken so that every small piece of it satisfies one of three criteria: it has zero weighted mean curvature, is at the end of the original edge, or it has a speed equal to the mobility times the undercooling at its ends which is the speed at which a zero-curvature edge would travel if it were located just off the end of the part. The routine is recursive so that every edge will break as much as possible.

With both of these methods, an edge is shattered only if it is longer than a certain pre-defined 'minimum edge length'. Thus varifolds are approximated by sequences of short sides having alternating normal directions. The choice of this minimum length is something of an art. Having it be too large can produce extra numerical noise, but if too small a value is chosen, the number of edges which are produced grows substantially. Most of these extra edges are very small, however, and if an edge is too small compared to the size of a single temperature field cell, then it is unable to accurately resolve temperature differences along its length, and so it does not increase to the accuracy of the computation (see Section 4.1.3). Thus in most of the computations shown here, the minimum edge length has been taken to be 1/4 of the distance across a grid cell. It is important to note that edges may become smaller than the 'minimum edge length' through the movement of adjacent edges, but they will not be shattered.

3.3.4 Heat release

The heat release step is designed to compute and place the latent heat produced or absorbed during crystallization or melting. It, like the heat flow step below, makes use of code written by Robert Almgren for the purpose [?]. When the computation is begun, an object χ_o of the field class is created to store an approximation to the characteristic function χ of the solid. The subroutine which produces this characteristic function loops through each edge of the interface and calculates exactly how much of the solid is contained in each grid cell of the field. These calculations are carried out in such a way as to preserve $\int \chi$, $\int x\chi$, $\int y\chi$, and $\int xy\chi$. Every time the interface moves, an object χ_n of the field class stores the new characteristic function of the crystal. We then add $L(\chi_n(i) - \chi_o(i))$ to each temperature cell $T(i)$ in the diffusion field (where L is the latent heat of fusion), increasing the temperature where new crystal has formed and lowering it where crystal has melted. The χ_n is then stored in place of χ_o for use in the next iteration.

3.3.5 Heat flow

Once a time step τ has been determined in the movement step, the heat equation must be solved on the temperature field for τ. Solving the heat equation on a rectangular grid is a standard and well studied problem. The method used is a straightforward explicit (or forward Euler) method [?]. At every time $t + \tau$ the new temperature $T(i, j, t + \tau)$ of a point (i, j) on the temperature grid is set at a weighted sum of $T(i\pm1, j, t)$, $T(i, j\pm1, t)$, and $T(i, j, t)$ itself. The coefficients used depend on the distance between grid cells in each direction. For the solution to be stable, the time step τ must be less than $\frac{1}{2D}(\Delta x)^2$, where D is the diffusion coefficient and Δx is the minimum distance across a grid cell. If τ is too large, several iterations of the solver are made with smaller time steps that sum to τ. This method vectorizes well, allowing an easy transition to a supercomputer if that becomes desirable, and it is possible to have different diffusivities in the solid and the liquid. The forward Euler scheme also can handle the non-smooth temperature distributions that result immediately after heat is added. As would be expected, for medium to large grids the heat flow step is the most time consuming.

4 Computational Results

In the computations discussed below, no particular efforts were taken to ensure that the parameters used fit any particular physical substance, however their physical meaning may be interpreted as follows. We wish to use units in which latent heat $L = 1$, heat capacity $c = 1$, and heat conductivity $k = 1$. Since there are four units of measurement that must be determined: computational length (l_c), energy (e_c), temperature (T_c), and time (t_c) units, we are allowed one degree of freedom once we know the physical parameters of the substance we wish to model. This degree of freedom essentially allows us to scale the system in some way. Using the values for nickel [?], for example, we find that $L = 2350 J/cm^3 = 1 e_c/{l_c}^3$, $c = 5.42 J/Kcm^3 = 1 e_c/T_c {l_c}^3$,$k = 0.155 cm^2/s = 1 {l_c}^2/t_c$. So $1T_c =$ 433K. Suppose we wish our computational box of size 10×10 to correspond to .001cm, then $1l_c = 0.0001$cm, $1e_c = 2.35 \times 10^{-9}$J, and $1t_c = 6.45 \times 10^{-8}$s. Our surface energy magnitude should be $\overline{\Phi} = 1.57 \times 10^{-4} e_c/{l_c}^2 = 3.7 \times 10^{-5}\text{J/cm}^2$.

4.1 Computational Validation

4.1.1 Grid Size

Perhaps the most obvious source of computational error is in the approximation of the smooth temperature field by a finite grid with bilinear interpolation between grid points. One of the effects of surface energy, however, is to make the interface stable to perturbations with small wavelength [?]. Thus, if the error produced by the computation grid is sufficiently small and local, it should have little effect on the accuracy of the computations.

Figure **??** show the same computation using grid sizes of 200 and 1000, with grid densities (i.e. grid points per unit length) of 20, and 100 respectively. The left computation shown in Figure **??** is a bit noisy: the sides of the dendrites are not completely smooth to the naked eye. In contrast, the right computation appears completely smooth. Figure **??** shows a plot of the tip speeds (averaged over the four arms) for several computations with different grid densities. The effect of grid density is more striking here. It can be seen that the tip velocities have settled down into a steady speed. Table **??** shows these steady-state velocities. As grid density increases, the tip speed decreases, however the steady-state velocities are within 1% of each other for grid densities of 80 and larger. Since tip speeds are quite sensitive to local variations, it seems reasonable to conclude that the computations are completely resolved with a grid density of 80.

The effect of variations in grid density is more dramatically shown in Figure **??**. In these calculations, the mobility is 80 and the undercooling is 0.7, producing a very rapid growth. The computation on the left, which has a grid density of 20, is extremely noisy, with a great deal of side branch activity. For a grid density of 80, however, there seems to be almost no noise or side branches at all.

4.1.2 Grid Induced Anisotropy

Although it seems that most low wavelength noise can be eliminated by using a sufficiently fine grid, interactions between the interface and the grid seems to produce some anisotropic effects even for relatively isotropic surface energies. Figure **??** shows a calculation in which the Wulff shape has 120 sides and the energy is the same in each Wulff direction. The initial seed was given a five-fold symmetric perturbation, so if the system were completely resolved, the resulting crystal should have an exact five-fold symmetry, but a careful examination shows that this is not the case. The effect of the underlying grid can be seen more strongly in Figure **??**, where the initial seed is round, but the grid size is 800×400: less dense in the y direction. The crystal shows a distinct tendency to grow in the vertical direction. The above computations show that dendrites grow faster in lower grid densities, however, so it is unclear that this demonstrates a bona fide grid anisotropic effect. Finally, Figure **??** shows a computation in which an initial seed having no imposed deformations grows on an 800×800 grid, but with the shattering routine disabled: no new edges are created. The grid has imposed an eightfold perturbation on it. Presumably the perturbation was first induced when the seed was very small, and was amplified via the Mullins-Sekerka instability [**?**].

4.1.3 The minimum edge length parameter

Faceted crystals growing in a diffusion field frequently form varifolds [**?**]. The computer program approximates these varifolds by a series of very small edges. In order to keep the total number of edges under control, the user can specify a 'minimum edge length', below which an edge will not shatter. Since this parameter is non-physical, we must ensure that it does not have a significant effect on the computation. Computations in which the minimum edge length is reduced show that although the number of edges in the computation increases, the morphology of the dendrites remains unchanged as long as the minimum edge length is smaller than the distance across a temperature grid cell. This is unsurprising since the temperature field cannot be resolved more finely than a temperature grid cell.

4.2 Simulation of Dendritic Growth

In this section we will examine some computational simulations of dendritic growth. In each case we began with a very small 'seed' in the center of the computational domain (which is generally a 10×10 box). The temperature of the whole box is set at some fixed undercooling, and the system is allowed to evolve.

4.2.1 Comparison with Ivantsov solutions

One of the few exact solutions to equations 1 and 2 holds in the limit as $M \to \infty$ and $\overline{\sigma} = 0$, i.e. $u(s) = 0$ on the interface [?]. These solutions are known as Ivantsov parabolas [?], because the solid is the set of points y such that

$$y \leq -x^2/2\rho + Vt \tag{5}$$

where ρ is the tip radius of curvature. If we let u_∞ be the undercooling as $y \to \infty$, then we can relate u_∞ to the *Peclet number*

$$\mathcal{P} = \frac{1}{2}\rho V \tag{6}$$

by

$$u_\infty = -\sqrt{\pi\mathcal{P}}e^{\mathcal{P}}\mathrm{erfc}(\sqrt{\mathcal{P}}) \tag{7}$$

where erfc is

$$\mathrm{erfc}(x) = 1 - \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt \tag{8}$$

Note that this relationship does not determine the tip radius and speed separately, but only their product. As might be expected, without surface energy, the speed is not uniquely determined. [?]

Comparison of simulations with Ivantsov solutions produces varying results, as is to be expected since rigorous Ivantsov solutions are based on several assumptions which do not hold true for most simulations using crystalline interfaces. The most obvious difference is that the Ivantsov parabola includes no surface energy effect, rather, the temperature at the interface is the melting temperature. Further, crystalline 'parabolas' are far from smooth at their tip, and the temperature can vary significantly along edges near the tip without inducing them to split. Thus, for many crystalline Wulff shapes, it is difficult to estimate the 'tip radius' for the dendrite. These concerns notwithstanding, it is possible to make some comparisons. In an effort to remove the effects both of surface energy and crystalline anisotropy, we ran several simulations using a Wulff shape having 120 sides, a side with normal direction θ having energy $\gamma(1 - \alpha\cos(4\theta))$. Thus we produce fairly smooth interfaces with four-fold anisotropy, the magnitude of the anisotropy depending on the parameter α. It is important to note that there are two 'types' of anisotropy at work here.

Because all computations use 'crystalline' surface energies, they are inherently anisotropic, but there is also the 'larger scale' anisotropy represented by α, and it will be shown that it is this anisotropy that controls the morphology of the growing dendrite. As would be hoped, for sufficiently large α, the small initial seed grows into a four armed crystal, each of which arms assumes a steady shape and growth rate. Figure **??** show several computations for which the energy multiplier is 0.001, the temperature is calculated on a grid density of 40×40, and the mobility is 100: they differ only in the anisotropy parameter α. Four distinct dendrites grow from the four corners though there is some side branching activity, especially for computations with low anisotropy. Figure **??** shows a plot of the dendritic tip velocities vs time for (bottom to top) $\alpha = 0.005, 0.007, 0.009$, and 0.01 (where we have averaged over the four arms), while Figure **??** shows the tip speeds (bottom to top) for $\alpha = 0.01, 0.02, 0.03$, 0.04, and 0.05. While somewhat noisy, they all do seem to be settling down to a fairly constant speed, though a slight upward curve may be detected for the smaller values of α suggesting that the steady operating state may not have been determined.

Concentrating on the calculation in which $\alpha = 0.01$, we can calculate the tip radius of each dendrite by performing a least-squares fit to the curves. For the values below, we fit the 40 edges nearest to the dendrite tip, however the calculated tip radius did not vary significantly with different numbers of points taken, indicating that the tip closely approximates a parabola. Figure **??** shows a plot of the Peclet number vs time after the dendrites have settled into the parabolic steady state. The four symbols represent the four dendritic tips. Note that the values for the four tips remain very close to one another, indicating that they are essentially the same calculation. The fluctuations appear random, but they are not very large, and they remain closely around the value of 0.187 predicted by equation 7 for an undercooling of 0.5. The time average of the Peclet numbers over the four tips is 0.185, which is extremely close to the predicted value.

As we change the parameters from the near isotropic, near zero surface energy, this fit to the predicted Ivantsov solution becomes less exact. In the fourth computation shown in Figure **??**, the anisotropy parameter α is 0.05. Note that the dendritic tips are much more clearly defined, with almost no side branch activity. The tip is less parabolic than in the above example: Figure **??** shows a plot of several Peclet numbers for which the 'tip radius' was determined by fitting parabolas to 20 data points at varying distance from the tip. The side of the dendrite closely fits a parabola with width 0.063, while the actual tip has much higher curvature than such a parabola. Using the plateau value of 0.063 as the 'tip radius', we find a Peclet number of 0.177, which is only 5% below the 'predicted' value of 0.187. It remains to be explained why a parabola fitted to the tip is so much sharper than one fitted along the side. Table **??** shows the mean value of the temperature along the dendrites integrated to various distances from the tip. Note that as the distance from the tip increases, the temperature at the interface more closely approximates 0. The Peclet number associated with an undercooling of 0.425 is 0.114, which is in the right range for parabolas near the tip. Table **??** shows the tip velocities, tip radii, and Peclet number for several calculations (including those shown in Figure **??**).

In contrast to the above calculations, Figure **??** uses a surface energy which is extremely anisotropic in both senses of the word. The Wulff shape is a truncated square, having magnitudes of 1 in the horizontal and vertical directions and $0.9\sqrt{2}$ in the diagonal directions (roughly corresponding to an α of .13). It is thus impossible for the tip to be at all smooth, though other parameters are similar to the above cases. The dendrites formed do approximate parabolas, however. In Figure **??** we again plot the Peclet number as determined from various distances along the edge, using the average tip speed of 7.2. Again, the dendrite seems to be much sharper near the tip than along the sides, approaching a plateau somewhere in the middle which is within 7% of the value predicted by equation 7. This seems quite reasonable since this example has begun to show a fair amount of side branch activity, which also would explain why the values seem to be somewhat high rather than low as was seen in the above case.

4.2.2 Effects of Variation of Surface Energy Magnitude

In the absence of side-branching, the effect of changes in the magnitude of the surface energy is somewhat subtle. Figure **??** compares calculations in which only the magnitude of the surface energy (equivalently, the capillary length) is changed. In the first one, some very short wavelength side branching can be observed. As would be predicted, the side branches disappear when the surface energy increases. At higher surface energies, the dendrites are blunter since edges must be much longer or temperature differences more severe in order to produce shattering. Figure **??** displays the average speed of the four dendritic tips with time for eight computations. Note that the tips take longer and longer to achieve a steady operating speed as energy increases, and it is not entirely clear that the last three have attained their final speed. In table **??** we display the number of edges, the operating speed (taken to be the tip speed near the end of the computations), the tip radius, and the Peclet number for each of the different energy multipliers. It should be noted, of course, that with stronger energy the tips become much less parabolic. Thus (particularly in the last two cases) the tip radius and Peclet values must not be given too much weight. Unsurprisingly, the Peclet number is close to the predicted value of 0.187 for low energies, but gets further away when the surface energy becomes high. Also for higher energies, the interface lags far behind the temperature field, which becomes much more spread out. This is why the computations ended relatively early, as they stop when the temperature field begins to affect the boundary.

4.3 Effect of Variations in the Mobility Magnitude

In this section we will examine the effect of changing the mobility parameter on the computation. Figures **??** shows 4 computations in which the undercooling and surface energy remain fixed, but the mobility varies from 1 to 64. Interestingly, the effect of mobility seems to be similar to that of surface energy magnitude. Table **??** shows the tip radius (ρ), tip speed, Peclet number, and number of edges for several computations. Note that in the first three computation, the tips have not reached a steady-state speed. The speed values for these computations are the final speeds attained.

In the first computation of Figure **??**, the mobility magnitude has the extremely low value of 1. The interface responds to the undercooling very slowly. Though the crystal shatters at an early stage, the breaks do not grow very quickly. Observation of the moving interface shows that the crystal seems to grow in a sort of layering fashion: once the corners have lifted up, short edges having zero curvature sweep in from the corners to meet in the centers of each edge. In the next computation, the corners of the crystal are forming distinct parabolas, and by the third, the tips are narrow enough to allow some points of negative curvature between tip and base.

A mobility of 32 produces a computation which matches a predicted Ivantsov solution, having a Peclet number close to 0.19. Higher mobilities have progressively larger Peclet values and also more side branch activity. In the last computation of Figure **??**, the sides of the dendrites seem somewhat rough and irregular, as if side branches may be about to sprout. This suggests that interfaces are more unstable at high speed. These computations also demonstrate the difficulty of producing 'clean' computations with high mobilities: small perturbations are exaggerated by the high mobility, and so the temperature grid must be made denser to compensate.

4.4 Movement using low temperature

As was mentioned in Section 3.3.1, it is possible to implement a different formula to determine the speed with which each edge should move. In particular, there is some physical justification to having the speed of each edge e be $v(e) = M(n(e))(-T(e) - \mathrm{wmc}(e))$ with $T(e)$ the low temperature along the edge. Small layers are assumed to nucleate at the coldest point and rapidly spread across the length of the edge. In order to shatter edges using this movement formula, we are obligated to use the first shattering method described in Section 3.3.3. The differences can be quite striking. Figure **??** shows two computations which use the trial and error shattering method. The computation on the left uses low temperature along the edge for movement. (Since we are interested in sidebranching, a somewhat coarse temperature grid was used). The resulting crystal is distinctly 'crystalline', having extremely angular dendrites and side branches. By comparison, the second computation, using the trial and error shattering and the mean temperature along an edge for movement, appears quite similar to computations that use the 'exact' shattering method, yet is far different from the computation on the left. It is less angular by far, and the dendrites are much more parabolic. This is perhaps to be expected, as the hypotheses to the Ivantsov parabolic solution (see section 4.2.1) are little respected by the computation in the first figure.

5 Conclusion

5.1 Evaluation of the method

A new method was presented for computing the motion of an interface under the influence of its curvature and a diffusion field. Since we restrict attention only to those interface energies which are completely crystalline, several computational problems are simplified. It is not required that curvature be calculated explicitly. Instead, we need only measure the lengths of edges. It is also easy to detect and make topological changes in the interface: since we know the allowed normal directions, we need only compare edges which have parallel normals to know if two parts of the crystal have intersected. Computations concerning the interface alone are also quite fast. Since the speed of the interface is explicitly calculated, we are only concerned with values along the interface itself. This is in contrast to the phase-field method, which must calculate the phase field at each point in the computational domain [**?**]. The speed of the computation is largely determined by the size of the temperature grid, since heat release and flow takes roughly 75% of the total computational time. It is also straightforward to change the exact formula used to determine the speed of the edges, the most difficult obstacle being determining a 'shattering routine' for the particular rule used. Preliminary computations suggest, however, that a general purpose scheme allowing all possible breaks to occur and finding which ones expand works reasonably well.

There are three limitations to the method all of which can be overcome to a greater or lesser degree. First, it must be able to approximate varifolds by many edges with small lengths. The minimum edge length parameter is used to keep these small edges from becoming unmanageably short, and it has been demonstrated (Section 4.1.3) that having such a parameter does not significantly affect the computations if it is chosen to be smaller than the size of a temperature grid cell. It is also possible to incorporate 'dummy' edge directions in the initial Wulff shape. Most of the computations shown above use a 'truncated square' Wulff shape, which is, in fact, an approximation to a true square. The extra diagonals allow dendritic arms to grow with varifold sides and many fewer edges. A related limitation is that the method can only be used to study completely crystalline surface energies. The program is able to handle Wulff shapes with many sides, however, so it is easy to approximate smooth surface energy functionals by crystalline ones with many sides. Finally, the mobility must be finite. In this it is similar to most competing methods (with the exception of [**?**]). It is possible to take a very high mobility in order to approximate an infinite mobility, but this has computational difficulties since it tends to exaggerate noise produced by the temperature grid.

It should be straightforward to model other moving boundary problems. Code has been written to include triple junctions in interfaces. This would allow such things as computations of grain growth and more realistic merging of boundaries when two seeds touch. The code has not been tested, however, and extra routines will be required to correctly model the motion of triple junctions.

It should be even easier to model the influence of multiple diffusion fields on the interface, a temperature field and a concentration field, for example, or concentrations of multiple materials. The exact shattering method is easily modified to handle two or more such fields. Multiple fields combined with triple junctions can be used to model eutectic growth. Vastly differing length and time scales of the diffusion fields or direct influences of one field on another would pose the same difficulties for this method as for any other, however.

It would also be desirable to study the effect of changing the velocity formula for edges. At this point the assumption has been made that speed is directly proportional to the driving force. It is possible, however, that velocity response to the driving force should not be simply linear, but exponential on facets around which the crystal is convex, and this would be simple to model using the crystalline computational model.

6 Acknowledgements

Thanks are due to Robert Almgren for the gracious use of his code and for many helpful conversations. This project would have been difficult to carry out without both. The authors are members of the Minimal Surfaces Team of the National Science and Technology Research Center for the Computation and Visualization of Geometric Structures in Minneapolis, supported by the National Science Foundation and the University of Minnesota, and several of the computations were performed on center machines. We wish to thank the staff of the center for their invaluable assistance. Thanks also to John Cahn, Craig Carter, and Jeff McFadden of NIST, and to Fred Almgren of Princeton University for critical reading of this manuscript.

The first author was supported by a Rutgers University Excellence Fellowship and the joint NIST/ARPA program on Computational Modeling of Microstructural Evolution in Advanced Alloy systems. The second author was partially supported by NSF-DMS91-05667.

A Tables

Table I: Steady-state tip speeds for different grid densities

Grid Density	Tip Speed
20	4.97
40	4.57
60	4.48
80	4.45
90	4.44
100	4.43

Table II: Average temperatures near tip (120-gon, $\alpha = 0.05$)

Edges from tip	Average temperature to tip
1	-0.075
8	-0.073
16	-0.058
32	-0.036
64	-0.022

Table III: Vital statistics for different anisotropies of a 120gon

α	ρ	Speed	$\mathcal{P}$
0.005	0.28	1.7	0.24
0.007	0.20	1.9	0.19
0.009	0.17	2.1	0.18
0.010	0.17	2.2	0.19
0.020	0.10	3.7	0.19
0.030	0.083	4.5	0.19
0.040	0.071	5.1	0.18
0.050	0.063	5.6	0.18

Table IV: Vital statistics for different capillary lengths

Emult	ρ	Speed	$\mathcal{P}$	Num Edges
0.0001	0.036	9.5	0.17	20364
0.0002	0.042	8.3	0.17	9808
0.0004	0.052	6.7	0.17	8700
0.0008	0.070	5.0	0.18	6972
0.0020	0.12	3.0	0.18	5240
0.0040	0.20	1.9	0.19	4920
0.0080	0.35	1.4	0.25	3512
0.0160	0.57	0.74	0.21	3024

Table V: Vital statistics for different mobility calculations

Mobility	ρ	Speed	$\mathcal{P}$	Num Edges
1	0.59	0.26	0.077	9784
2	0.59	0.40	0.12	12552
4	0.38	0.70	0.13	21144
8	0.26	1.2	0.16	26696
32	0.11	3.3	0.18	12894
64	0.07	5.2	0.18	4546
128	0.051	7.6	0.19	2798

B Figures