LBNL-39216
UC-405

ERNEST ORLANDO LAWRENCE
BERKELEY NATIONAL LABORATORY

An Adaptive Level Set Method

R.B. Milne
Physics Division
Mathematics Department

December 1995
Ph.D. Thesis

DISCLAIMER

LBNL-39216

AN ADAPTIVE LEVEL SET METHOD*

Roger Brent Milne
Lawrence Berkeley National Laboratory
University of California
Berkeley, CA 94720

Ph.D. Thesis

December 1995

* This work was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract Number DE-AC03-76SF00098.

Abstract

An Adaptive Level Set Method

by

Roger Brent Milne
Doctor of Philosophy in Mathematics

University of California at Berkeley

James A. Sethian, Chair

This thesis describes a new method for the numerical solution of partial differential equations of the parabolic type on an adaptively refined mesh in two or more spatial dimensions. The method is motivated and developed in the context of the level set formulation for the curvature dependent propagation of surfaces in three dimensions. In that setting, it realizes the multiple advantages of decreased computational effort, localized accuracy enhancement, and compatibility with problems containing a range of length scales.

To Krista

Contents

Acknowledgements

The production of this thesis would not have been possible without the contributions of many people, all of whom have my deepest gratitude. Most of all, I would like to thank my advisor, James Sethian, for the unfaltering support and the generous advice he has given to me over the past several years. It has been both a pleasure and a privilege to have worked with him. I am also grateful to the other members of my committee, Alexandre Chorin and David Culler, for their reading of this thesis and their helpful suggestions.

Finally, I would like to acknowledge the financial support of my graduate studies through the National Science Foundation and the Mathematics Group at the Lawrence Berkeley National Laboratory. All of the computations cited in this thesis were carried out at the Lawrence Berkeley National Laboratory.

Chapter 1

The Evolution of Surfaces

This thesis develops a particular numerical technique to model evolving surfaces. In this chapter, the concepts of a surface and its evolution are formalized, and some relevant theoretical results are relayed. The goal is to lay a framework for the numerical constructions presented in later chapters, and the material has been specifically chosen for its applicability in that limited context.

Evolving surfaces appear in an broad variety of problems, ranging across fields as diverse as material science and medical imaging [39]. In these problems, the surface is often specified as the boundary between two material regions which change in time, and the evolution of the interface is dictated by their motion. A prototypical example is given by the process of solidification. Imagine a space filled with a single material in its solid and liquid states, and let the region occupied by the solid be bounded and simply connected. The evolving surface is then the boundary of this region and its evolution is given by the growth of this region as solidification occurs (or its reduction through liquefaction).

More specifically, given an initial surface Γ_0 and temperature field u_0, the evolution of the system can be modeled as follows. The heat equation holds for u in each region, and as the surface moves with a normal velocity proportional to the jump in the derivative of u across it, it acts as a heat source (through the release of latent heat). This is the classical Stefan Problem.

The behavior becomes more complex in the special case where the liquid has initially been cooled to a temperature below the freezing point. In the absence of any solid, it is possible for a liquid to exist in this metastable, undercooled state, but once a solid seed of sufficient size is introduced, the system becomes unstable and the solid will quickly

grow into the melt in a dendritic fashion as illustrated in Fig–1.1. In this case, application of the above model would result in unbounded velocities for the dendritic tips. As a tip extended into the melt it would encounter a growing jump in $\partial_{\mathbf{n}} u$, and its velocity would increase exponentially. To eliminate this instability, a modification to the model is required. The stabilizing effect that needs to be included is that of surface tension which results in a decrease in velocity proportional to the local (mean) curvature of the boundary. With this modification the development of sharply pointed dendrites is slowed and the resultant velocity is bounded. This is the Modified Stefan Problem; see [11, 13, 20, 21, 24, 40] for thorough discussions of this problem.

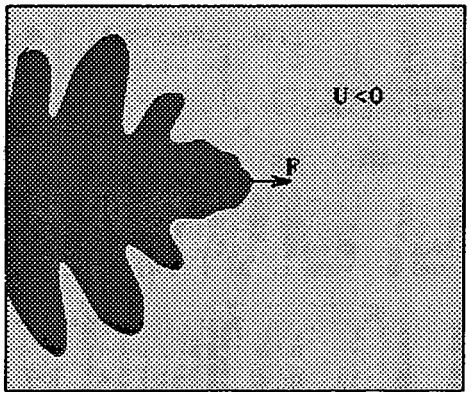


Figure 1.1: Dendritic Solidification

The role of surface tension here — and its bearing on stability — is representative of a much wider class of models in which the evolution of the surface includes a curvature dependent term. The incorporation of curvature into the model will be given particularly strong emphasis in the analytical and numerical discussions that follow both because of its fundamental importance and because of the difficulty in numerically approximating its influence.

1.1 Formulations for Surfaces

To develop a mathematical model for an evolving hypersurface, a formulation for the surface itself must first be laid out. There are several different ways of specifying a surface that will prove useful in later discussions; each of these descriptions provides a distinct range of information about the surface.

From one perspective, the implicit definition used in the previous section could be formalized, and Γ defined as the boundary (∂U) of an open set $U \subset \mathbb{R}^n$. Thus, Γ would "separate" the two regions, U and $\mathbb{R}^n \setminus \overline{U}$. An example is shown in Fig–1.2. The defining region U need not be bounded, nor need it be connected. In fact, the ability to naturally represent disconnected regions and the evolution from a connected region to disconnected ones (and *vice versa*) is a strength of this formulation. At the same time, however, writing characteristics of the surface (such as the curvature) in terms of the set U is a cumbersome proposition.

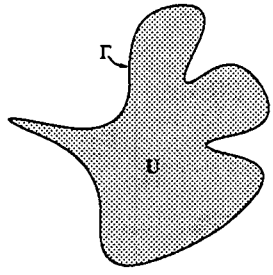


Figure 1.2: Surface as the Boundary of a Set

Alternatively, Γ could be given as a parameterization, i.e. as the range of a continuous map from an object (often a manifold) of dimension $n-1$ into $\mathbb{R}^n$(See Fig–1.3):

$$\Gamma = \{x = p(s) \in \mathbb{R}^n : s \in \mathbf{M}^{n-1}\}.$$

With this formulation the advantages and disadvantages of the former are neatly reversed; namely, surface characteristics are easily established, but a topological change from one region to two is problematic (involving, typically, a change in the parameterization space).

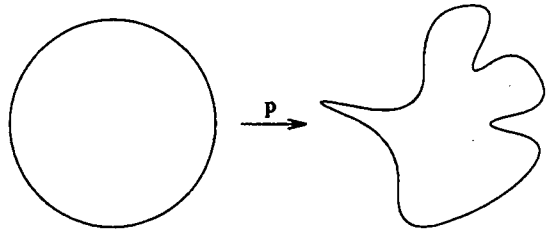


Figure 1.3: Surface as a Parameterization

Finally, the formulation that underlies the numerical scheme discussed in this work

is the definition the surface as the (zero) level set of of a continuous scalar function on $\mathbb{R}^n$:

$$\Gamma = \{\mathbf{x} \in \mathbb{R}^n : \varphi(x) = 0\}$$
$$\varphi : \mathbb{R}^n \to \mathbb{R}$$

An illustration is given in Fig–1.4. Superficially, this formulation would seem the most artificial of the three, but in practice it is both natural and powerful, maintaining the separate advantages of both of the foregoing descriptions. It provides a natural setting for topological change equal to that of the first formulation — in fact, it can be considered as a generalization of that formulation with $\varphi < 0$ in U and $\varphi > 0$ in $\mathbb{R}^n \setminus \overline{U}$. At the same

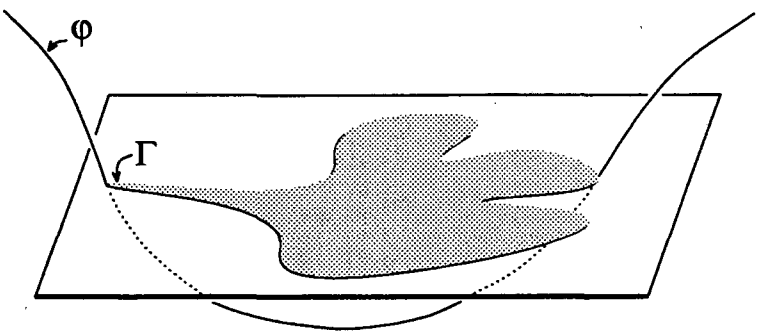


Figure 1.4: Surface as a Zero Level Set

time, it allows characteristics of the surface to be conveniently written in terms of φ, and in that way rivals the description of the surface as a parameterization. For example, the unit normal to the surface is immediately established (assuming differentiability) by the gradient of φ:

$$\hat{\mathbf{n}} = \frac{\nabla\varphi}{|\nabla\varphi|}. \tag{1.1}$$

From a numerical perspective, this formulation will therefore prove particularly appealing, as is underscored by the fact that this view of a surface and the accompanying methodology were developed by Osher and Sethian in the numerical context of [31].

1.2 Propagation of Surfaces

Given an initial surface Γ_0, the evolution of the surface, Γ_t, is constructed as follows. Let $\hat{\mathbf{n}}(x, \Gamma_t)$ be the unit normal vector to Γ_t at the point $x \in \Gamma_t$. For these normals to exist, a certain degree of smoothness is required of Γ_t; it will be assumed that Γ_0 has a

well defined normal direction at every point, and the solution will be said to exist only as long as Γ_t retains this degree of smoothness, say for $0 \leq t < t_c$. Eventually, using the level set formulation, it will be possible to generalize the solution so that it can be continued past the time at which the normal ceases to exist at some points. But for now, the classical notion of propagation under a speed function $F(\mathbf{x}, t, \Gamma_t)$ can be given as:

$$\Gamma_t = \{X(x,t) : x \in \Gamma_0\} \tag{1.2}$$
$$\dot{X}(x,t) \;=\; F(X,t,\Gamma_t)\,\hat{\mathrm{n}}(X,\Gamma_t) \qquad\qquad X(x,0) = x \quad .$$

That is to say, Γ_t consists of all points following the trajectories given by the vector field $F\,\hat{\mathrm{n}}$ and originating from some point of Γ_0. See Fig–1.5. Note that F has been written as depending on space and time (perhaps the temperature there) and also on the surface itself (perhaps its curvature).

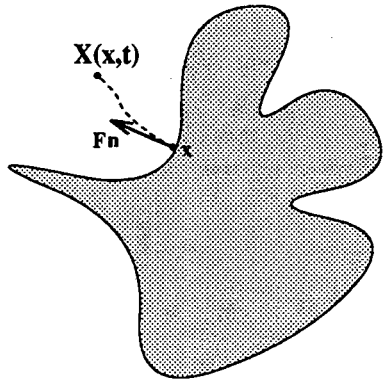


Figure 1.5: An Evolving Surface

For the level set style of formulation, this construction of Γ_t can be carried over immediately by extending φ to be a function of time and simply requiring that the zero level surface of φ always coincide with Γ_t:

$$\{\mathbf{x} \in \mathbf{R}^n : \varphi(x,t) = 0\} = \Gamma_t\,.$$

Then by substituting the definition of Γ_t from Eq–1.2:

$$\varphi\,(X(\mathbf{x},t),t) = 0 \quad \Leftrightarrow \quad x \in \Gamma_t\,,$$

and differentiating with respect to time:

$$\varphi_t + \nabla\varphi(X,t)\cdot\dot{X} \;=\; 0$$

$$\begin{aligned} \varphi_t + F\,\nabla\varphi(X,t)\cdot\hat{\mathrm{n}}(X,\Gamma_t) &= 0 \\ \varphi_t + F\,\nabla\varphi(X,t)\cdot\frac{\varphi(X,\Gamma_t)}{|\varphi(X,\Gamma_t)|} &= 0\,, \end{aligned}$$

an evolution equation for φ is obtained:

$$\varphi_t + F|\nabla\varphi| = 0\,, \tag{1.3}$$

with the initial condition supplied by a choice of φ_0 that is continuous and that has the appropriate zero level set.

As has already been alluded to, an important class of propagation is that of surfaces moving according to their (local) mean curvature, $F = F(\kappa)$. Once again, the curvature proves an easy object to compute for the level surfaces of φ, since for any vector field $\mathbf{v}$ normal to a surface, so the curvature is given by (see [41]):

$$\kappa = \nabla\cdot\left(\frac{\mathbf{v}}{|\mathbf{v}|}\right),$$

and, therefore,

$$\kappa = \nabla\cdot\left(\frac{\nabla\varphi}{|\nabla\varphi|}\right) = \frac{\sum_i\sum_{j\neq i}(\varphi^2_{x_j}\varphi_{x_ix_i} - \varphi_{x_i}\varphi_{x_j}\varphi_{x_ix_j})}{(\sum_i\varphi^2_{x_i})^{3/2}}. \tag{1.4}$$

Finally, for the case of collapse under mean curvature, $F = -\kappa$, substituting Eq–1.4 into Eq–1.3 gives:

$$\varphi_t = \frac{\sum_i\sum_{j\neq i}(\varphi^2_{x_j}\varphi_{x_ix_i} - \varphi_{x_i}\varphi_{x_j}\varphi_{x_ix_j})}{(\sum_i\varphi^2_{x_i})^{1/2}}. \tag{1.5}$$

This equation has been extracted from the classical notion of a propagating surface, but the correlation between the two is not entirely straight forward. First, there is the curious choice of an arbitrary φ_0, subject only to the constraints that it be continuous and have a zero level set corresponding to Γ_0. Then there is the more disturbing fact that the evolution equation is only well defined where $\nabla\varphi \neq 0$. The next section will address some of these concerns and outline the theoretical background that will also prove helpful in the analysis of certain related numerical issues.

1.3 Theoretical Background

The foregoing has all been laid out under the stipulation that the surface remain reasonably smooth, i.e. that no corner develops at which the normal (or the other relevant

surface characteristic such as curvature) become undefined. Even for primitive speed functions, such corners can arise — and do in the prototypical case of $F = 1$, with an initial surface (curve) given by the graph of cosine:

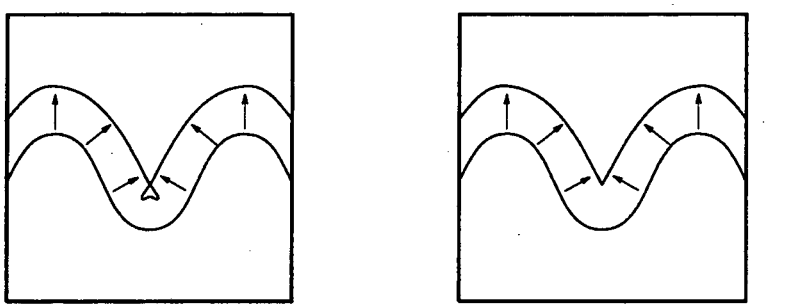
$$\Gamma_0 = \{(x, y) : y = \cos(x)\} \subset \mathbb{R}^2 ,$$

for which corners will develop in finite time [35]. For this particular problem with its constant speed function, one way of extending the solution would be to establish the "particle" trajectories based solely on the normals to the initial surface:

$$\Gamma_t = \{X(x,t) : x \in \Gamma_0\}$$

$$\dot{X}(x,t) = \hat{\mathrm{n}}(x, \Gamma_0), \quad X(x,0) = x ,$$

but this will lead to the "swallow tail" solution shown in Fig–1.6a, which is unacceptable if the surface is meant to represent the boundary between two region with one propagating into the other. If, for example, the surface represents a flame front moving into a region of unburnt gas and leaving in its wake a region of burnt gas, then the natural entropy condition given by Sethian in [34] is that a point of burnt gas remain burnt for all time. The resulting entropy satisfying solution corresponds to Huygen's construction in which Γ_t is composed of all points of distance t from Γ_0 (Fig–1.6b). Furthermore, this solution can also be shown to be the limit of of the viscous problem $F = 1 - \varepsilon\kappa$ (under which the curve will remain smooth for all time), with the limit taken as $\varepsilon \to 0$ [35].



(a) Swallow-Tail Solution (b) Entropy Satisfying Solution

Figure 1.6: Two Solutions for A Propagating Curve

For more complex speed functions, the issues of existence and uniqueness also become more involved. Such is the case for the classical problem of collapse under mean curvature. In two dimensions, it is known that any bounded, closed, smooth curve collapses

under curvature smoothly to a point [19]. In three dimensions, however, this is no longer the case (for surfaces), the classic counter-example [36] being a dumbbell shaped surface with a long, thin "bar" that pinches off in the center in finite time as it moves under mean curvature. As the pinching occurs, the curvature becomes infinite, corners are formed, and the continuance of the classical solution ceases.

By framing the evolution in the level set modality (Eq–1.5), the problem is generalized in such a way that the solution can be continued in a sensible fashion past such critical times. This generalization was analyzed by Evans and Spruck in a series of papers [14, 15, 16, 17] to which the interested reader is referred for the details; here, only the relevant results are summarized. Most importantly, they were able to construct a weak solution to Eq–1.5 and guarantee its existence and uniqueness for all time. Furthermore, the motion of the zero level surface of this weak solution was shown to be identical to that given by the classical notion of collapse under mean curvature, so long as the latter exists. It is therefore reasonable to call Eq–1.5 a generalized evolution by mean curvature.

Since the motion of the zero level set cannot then depend on the original choice of φ_0 (continuous and with a fixed zero level set), it follows that perturbing φ away from its zero level set can have no effect on the evolution of that surface. This fact can be restated more generally if it is first noted that there is nothing special about the zero level set and that every level set of φ obeys the same evolution equation; that is Eq–1.5 gives the evolution of a family of surfaces collapsing under curvature. It then follows that a perturbation of some collection of level surfaces of φ will never effect any of its other level surfaces — put figuratively, there can be no transmission of information across the level surfaces of φ.

Conversely, Evans and Spruck were able to demonstrate that "information" travels infinitely fast (instantaneously) along a given surface. Specifically, they showed that if Γ' and Γ are the smooth boundaries of the bounded and connected open sets $U' \subset U \subset \mathbb{R}^n$ with $U' \neq U$, then $\Gamma'^t \cap \Gamma^t = \emptyset \ \ \forall t > 0$. So the two surfaces "tear apart" instantly, regardless of the extent of their initial intersection; see Fig–1.7. Such effects are characteristic of parabolic partial differential equations, and indeed the evolution equation for φ was shown to be uniformly parabolic along the level surfaces, even while it is degenerate across them. A heuristic interpretation of Eq–1.5 is therefore as a dispersion equation, where it is curvature that is being dispersed and dispersed only along the level surfaces.

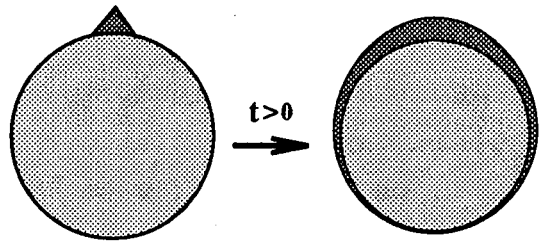


Figure 1.7: "Tearing Apart" of Surfaces under Curvature Flow

1.4 An Example

A useful example, both as a concrete demonstration of the preceding derivation and as a basis for some of the numerical work to come, is that of spheres propagating under the speed function:

$$F = a - b\kappa\,,$$

where a and b are constants, with b positive. (The evolution is only stable for $b > 0$, for a negative b the equation is akin to the backwards heat equation; see [31].)

Denoting the center of the spheres as $\mathbf{x_0}$, the evolution is spherically symmetric in $r = |\mathbf{x} - \mathbf{x_0}|$ with φ satisfying the PDE:

$$\varphi_t(r,t) + (a - \frac{b}{r})\varphi_r(r,t) = 0$$

whose solution is given by

$$\varphi\,(R(t),t) = \varphi\,(R(0),0) \qquad \text{where} \quad R_t = \frac{b}{R} - a\,.$$

For explicitness, the initial condition is taken to be the signed distance function to a sphere of some radius R_0, also centered about $\mathbf{x_0}$:

$$\varphi(r,0) = r - R_0\,.$$

For the two special cases of $a = 0$ and $b = 0$, the ODE for R is separable and can be solved as:

$$R(t) = R(0) - at \qquad \Rightarrow \qquad \varphi(r,t) = r - (R_0 + at) \qquad \text{for} \quad b = 0$$

$$R^2(t) = R^2(0) - 2bt \qquad \Rightarrow \qquad \varphi(r,t) = \sqrt{r^2 + 2bt} - R_0 \qquad \text{for} \quad a = 0$$

For the general problem with $a \neq 0$, $b \neq 0$, the ODE can be made exact through multiplication of the integrating factor $e^{-\frac{a^2}{b}t}e^{\frac{a}{b}R}$, and then solved to produce an implicit equation

relating R(0) and R(t):

$$(aR(t)-b)\,e^{-\frac{a^2}{b}t}e^{\frac{a}{b}R} \;=\; (aR(0)-b)\,e^{\frac{a}{b}R(0)}$$

$$\Rightarrow \quad b\ln(aR(t)-b) - b\ln(aR(0)-b) + a\,(aR(t)-aR(t)) \;=\; a^2t\,.$$

Using this equation, $\varphi(r,t)$ can be constructed by solving for $R(0)$ with $R(t)=r$, and then setting

$$\varphi(r,t) = \varphi\,(R(0),t) = R(0) - R_0\,.$$

Chapter 2

Numerical Methods For Evolving Surfaces

Each of the three ways of describing surfaces given in §1.1 corresponds to a class of numerical methods for tracking the evolution of a surface. The intrinsic strengths and weaknesses of each description will be seen to reemerge in the numerical methods it inspires. This chapter briefly recounts the formulation and critiques the performance of each of these classes of methods. It then delves into the numerical methods for the level set equation in detail, as they will become the focus of the following chapters.

2.1 A Survey of Methods

The first methods used to track the evolution of a surface were based on parameterizations. Such methods come in a variety of flavors, but a representative example is that of "marker particles" in which a collection of points (the particles) are spaced across the surface and then moved along trajectories according to the speed function F.

The surface at a time t can then be reconstructed from the positions of the marker particles at that time. A primary deficiency of such methods is in their inability to naturally handle topological changes (just as is the case for the representation of surfaces as parameterized objects). If the surface is intended to follow the boundary of two expanding regions which eventually flow together and form a single region, then a change of parameterization will have to be made when they join, and the particles in the region of contact between the two surfaces will have to be identified and removed.

Similar problems develop even in the motion of a single surface. Where the trajectories flow together (as in Fig–1.6) and corners form, a marker particle scheme leads to two difficulties. First, they naturally produce the swallow-tail solution of Fig–1.6 as each particle blindly follows its locally determined trajectory (independently of its crossing of other trajectories). Since this is usually not the desired entropy satisfying solution, the "tails" must be removed in a process called de-looping [22]. Secondly, the markers will accumulate at these corners, and if unchecked, the breakdown in their spacing will lead to instability in the difference operators based on their positions [31, 35].

These difficulties are avoided in an entirely different class of techniques know as "volume of fluid" methods; these were motivated by a desire to track multiple fluids and introduced by Noh and Woodward in [30]. They are based on the representation of a surface as the boundary of a region, and as such they can naturally address the above issues of topological change and entropy satisfaction. Indeed, the behavior that has been demanded of the marker particle scheme is that as envisioned for the surface of a material region — as is typically the case in a physically motivated process. Volume of fluid methods intrinsically exhibit such behavior. They work by discretizing the volume of U as a percentage of each cell. The surface can then be reconstructed in the cells containing a partial volume of U (neither zero nor one) and the motion of the surface tracked through its effect on the nearby volumes.

In practice, the reconstruction is not carried out explicitly, but it is used to derive the way in which the discretized volume is updated. When the motion depends on geometrical information such as the normal direction or curvature of the surface, the derivation becomes very complicated and the accuracy of the method difficult to insure [10, 23, 32].

Using the level set equation as the basis for a numerical scheme for tracking surfaces offers many of the same advantages as the volume of fluid method. Furthermore, it also allows computation of geometrical quantities in natural forms. In fact, the formulas given in §1.1 and §1.2 for the normal and the curvature will be carried over directly into the numerical schemes. This combination of features make the level set method particularly appealing, and its implementation will now be considered in some detail.

2.2 The Level Set Method

The level set method amounts to the numerical solution of the PDE given in Eq–1.3 with the speed function, F, possibly depending on on the derivatives of φ because of the effect of local surface characteristics. For the remainder of this work, the dependence of the speed function on the surface will be restricted to the effects of curvature alone, $F = F(\mathbf{x}, t, \kappa)$, and that dependence further restricted to be a linear one. Other inclusions can easily be made, such as dependence on the direction of the gradient (to model a spatial anisotropy [40]), but the essential elements of the method are illustrated by $F = F(\mathbf{x}, t, \kappa) = a(\mathbf{x}, t) - b(\mathbf{x}, t)\kappa$.

The first term is called the "hyperbolic term," as Eq–1.3 becomes a hyperbolic PDE for $F = a(\mathbf{x}, t)$. On the other hand, for $F = b(\mathbf{x}, t)\kappa$ Eq–1.3 becomes Eq–1.5 (plus the addition of the coefficient b) which as already mentioned is a parabolic equation along each level surface, and the second term is therefore called the parabolic term. These two terms require separate numerical handling and Eq–1.3 is rewritten as:

$$\varphi_t = -F_a|\nabla\varphi| - F_b|\nabla\varphi|\,, \tag{2.1}$$

and the two contributions to φ_t are computed separately and then combined. Explicit methods are used for both terms, and the time stepping is carried out by a forward Euler method. This is the simplest possible construction. A more elaborate time stepping method could easily be included, but the simpler case suffices to illustrate the methods that follow.

2.3 The Hyperbolic Term

This section outlines the development, as given by Osher and Sethian in [31], of a first order "upwinding" scheme for approximating the right hand side of the hyperbolic equation

$$\varphi_t = -a(\mathbf{x}, t)|\nabla\varphi|\,.$$

It is instructive to first consider the one dimensional variation: $\varphi_t = -a(x, t)|\varphi_x|$, since this is a special case of the Hamilton-Jacobi equation

$$\varphi_t = -H(\varphi_x) \qquad \text{with} \qquad H = a(x, t)\sqrt{\varphi_x^2}\,.$$

Differenting with respect to x and substituting $u = \varphi_x$ transforms the evolution equation into the hyperbolic conservation law

$$u_t = -H(u)_x\,,$$

which can be approximated by the conservative numerical scheme established by

$$(D_t u)_i = \frac{1}{\Delta x}(g_{i-1/2}(u) - g_{i+1/2}(u))\,,$$

where g is an appropriate numerical "flux function." Then by integration, a scheme for φ is obtained:

$$D_t\varphi = g(D_{\pm x}\varphi)\,.$$

For the special case of $H(\varphi_x) = a\sqrt{\varphi_x^2}$, the use of the following numerical flux function completes the description of the upwinding scheme:

$$g(D\varphi) = \begin{cases} a(x,t)\Big(\min^2(D_{-x}\varphi,0) + \max^2(D_{+x}\varphi,0)\Big)^{\frac{1}{2}} & a(x,t) \geq 0 \\ a(x,t)\Big(\min^2(D_{+x}\varphi,0) + \max^2(D_{-x}\varphi,0)\Big)^{\frac{1}{2}} & a(x,t) \leq 0 \end{cases}$$

In higher dimensions this trick can no longer be employed to recast the equation as a hyperbolic conservation law. Nonetheless, a similar scheme for φ still holds, and in three dimensions, where $H(\varphi_x,\varphi_y,\varphi_z) = a\sqrt{\varphi_x^2+\varphi_y^2+\varphi_z^2}$, is provided by the following approximation:

$$D_t\varphi = g(D_{\pm x}\varphi, D_{\pm y}\varphi, D_{\pm z}\varphi)\,, \tag{2.2}$$

under a numerical flux now given by

$$g(D\varphi) = \begin{cases} a(\mathbf{x},t)\sqrt{\begin{array}{l}\min^2(D_{-x}\varphi,0) + \max^2(D_{+x}\varphi,0) \\ +\min^2(D_{-y}\varphi,0) + \max^2(D_{+y}\varphi,0) \\ +\min^2(D_{-z}\varphi,0) + \max^2(D_{+z}\varphi,0)\end{array}} & a(\mathbf{x},t) \geq 0 \\ a(\mathbf{x},t)\sqrt{\begin{array}{l}\min^2(D_{+x}\varphi,0) + \max^2(D_{-x}\varphi,0) \\ +\min^2(D_{+y}\varphi,0) + \max^2(D_{-y}\varphi,0) \\ +\min^2(D_{+z}\varphi,0) + \max^2(D_{-z}\varphi,0)\end{array}} & a(\mathbf{x},t) \leq 0 \end{cases}$$

Higher order extensions to this scheme can also be found in [31].

2.4 The Parabolic Term

Osher and Sethian also provide (in [31]) a scheme for the parabolic term which will also be the basis of the scheme used here. In it, given the parabolic nature of Eq–1.5, each of the spatial derivatives are replaced by their central finite difference equivalents (just as would be done for the heat equation), and the resulting finite difference approximation of the time derivative is

$$D_t\varphi = \frac{\begin{array}{c}(D_y^2\varphi + D_z^2\varphi)D_{xx}\varphi + (D_x^2\varphi + D_z^2\varphi)D_{yy}\varphi + (D_x^2\varphi + D_y^2\varphi)D_{zz}\varphi \\ -2D_y\varphi D_z\varphi D_{yz}\varphi - 2D_x\varphi D_z\varphi D_{xz}\varphi - 2D_x\varphi D_y\varphi D_{xy}\varphi\end{array}}{(D_x^2\varphi + D_y^2\varphi + D_z^2\varphi)^{1/2}}\,. \tag{2.3}$$

2.5 Time Step Restrictions

Given the approximations for φ_t above, the forward Euler step can be applied, but the size of its time step, Δt, also needs to be established. For a method on a uniform grid, trial and error usually suffice in this regard, but for the adaptive computations to come it will be important that the proper time step can be approximated for the above methods. As each of the methods contributes its own restriction to the size of the time step, they are considered here in turn.

For the three-dimensional problem, let the spacings of the numerical grid be given by Δx, Δy, and Δz. Then the approximation of the hyperbolic term as in §2.3 leads to a CFL restriction on Δt (again, see [31]):

$$\Delta t \leq \frac{|\nabla\varphi|}{2|a|\,(\frac{|\varphi_x|}{\Delta x} + \frac{|\varphi_y|}{\Delta y} + \frac{|\varphi_z|}{\Delta z})}, \tag{2.4}$$

with the derivatives of φ being evaluated as the finite differences appearing in Eq–2.2, and the timestep then taken as a minimum across the domain.

The method for the parabolic term given in §2.4 typically places a more stringent requirement on Δt, which, as it has been modeled on the explicit method for the heat equation, can be expected to be of $\mathcal{O}(\Delta x^2)$. This is a stability condition on Δt which is approximated here for the first time through a stability analysis of the linearized version of the forward Euler method following from Eq–2.3:

$$\varphi \leftarrow \varphi + \Delta t(a_1 D_{xx}\varphi + a_2 D_{yy}\varphi + a_3 D_{zz}\varphi + c_{23} D_{yz}\varphi + c_{13} D_{xz}\varphi + c_{12} D_{xy}\varphi)\,.$$

The amplification factor, ρ, of this scheme is then given by:

$$\begin{aligned}\rho = 1 \quad &+ \quad 2\alpha_1(\cos\theta_1 - 1) + 2\alpha_2(\cos\theta_2 - 1) + 2\alpha_3(\cos\theta_3 - 1) \\ &- \quad \gamma_{23}\sin\theta_2\sin\theta_3 - \gamma_{13}\sin\theta_1\sin\theta_3 - \gamma_{12}\sin\theta_1\sin\theta_2\,,\end{aligned}$$

where

$$\alpha_i = a_i\frac{\Delta t}{\Delta {x_i}^2} \qquad \gamma_{ij} = c_{ij}\frac{\Delta t}{\Delta x_i \Delta x_j}\,,$$

and the (linearized) scheme will be stable as long as $|\rho| \leq 1$. The condition that has a bearing on the time step is $\rho \geq -1$, as if ρ is ever larger than one, the scheme will be unstable regardless of the time step. In the general linear problem, this can be the case if $a_i < 0$ (the backwards heat equation) or if $a_1 = a_2 = 0$ and $c_{12} > 0$, as taking $(\theta_1, \theta_2, \theta_3) = (\frac{\pi}{2}, \frac{\pi}{2}, 0)$ gives $\rho = 1 + \gamma_{12}$. For the purpose of computing an appropriate time step, these cases can disregarded (in fact, for the "coefficients" arising from the curvature term in Eq–2.3, they cannot occur) and only $\rho \geq -1$ considered.

A formula for the extrema of ρ (as a function of the θ's) would then determine the maximum stable time step. A closed analytic expression for its extrema could not be found; a bound was, however, successfully constructed by first re-arranging and grouping the terms:

$$\begin{aligned}\rho = 1 \quad &+ \quad \underbrace{\alpha_1(\cos\theta_1 - 1) + \alpha_2(\cos\theta_2 - 1) - \gamma_{12}\sin\theta_1\sin\theta_2}_{f_3(\theta_1,\theta_2)} \\ &+ \quad \underbrace{\alpha_1(\cos\theta_1 - 1) + \alpha_3(\cos\theta_3 - 1) - \gamma_{13}\sin\theta_1\sin\theta_3}_{f_2(\theta_1,\theta_3)} \\ &+ \quad \underbrace{\alpha_2(\cos\theta_2 - 1) + \alpha_3(\cos\theta_3 - 1) - \gamma_{23}\sin\theta_2\sin\theta_3}_{f_1(\theta_2,\theta_3)}\,.\end{aligned}$$

Writing then the class of equations of which the above f_i's are members as

$$f(x,y) = a(\cos x - 1) + b(\cos y - 1) + c\sin x\sin y\,,$$

the bound on Δt can then be established from an expression for the extrema of f in terms of its coefficients, since

$$\rho \geq -1 \quad \text{and} \quad \rho \geq 1 - 3|f_{ext}|\Delta t \qquad \Rightarrow \qquad \Delta t \leq \frac{2}{3|f_{ext}|}\,. \tag{2.5}$$

At the extrema of f, x and y satisfy the following non-linear system of equations:

$$\begin{aligned} a\sin x + c\sin y\cos x &= 0 \\ b\sin y + c\sin x\cos y &= 0\,, \end{aligned}$$

for which one solution is $\sin x = \sin y = 0$ (which is the familiar and only solution for $c = 0$) providing the following candidates for extrema:

$$f_{ext} = a(\pm 1 - 1) + b(\pm 1 - 1)$$

$$\text{(Eq–2.5)} \quad \Rightarrow \quad \Delta t \leq \frac{1}{3|a| + 3|b|} . \tag{2.6}$$

For the other extrema, it can then be assumed that $\sin x$, $\sin y$, and c are all non-zero, and the system of equations can be re-written as:

$$\cos^2 x = \frac{a^2 \sin^2 x}{c^2 \sin^2 y} \qquad \cos^2 y = \frac{b^2 \sin^2 y}{c^2 \sin^2 x}$$

which leads (trough ample application of the Pythagorean Theorem) to the following expression for the extrema of f:

$$f_{ext} = a\left(\pm\frac{a}{c}\sqrt{\frac{b^2+c^2}{a^2+c^2}} - 1\right) + b\left(\pm\frac{b}{c}\sqrt{\frac{a^2+c^2}{b^2+c^2}} - 1\right) \pm \frac{c^4 - a^2b^2}{c\sqrt{a^2+c^2}\sqrt{b^2+c^2}} .$$

And setting

$$s_1 = \sqrt{a^2 + c^2} \qquad s2 = \sqrt{b^2 + c^2} ,$$

the restriction from Eq–2.5 can be written as:

$$\Delta t \leq \frac{2/3}{|a| + |\frac{a^2}{c}|\frac{s_2}{s_1} + |b| + |\frac{b^2}{c}|\frac{s_1}{s_2} + |\frac{c^2 - a^2b^2}{c}|\frac{1}{s_1 s_2}} . \tag{2.7}$$

The time step requirement is therefore bounded by the minimal value obtained by Eq–2.6 and Eq–2.7 over all choices of:

$$a = \frac{a_i}{{\Delta x_i}^2} \qquad b = \frac{a_j}{{\Delta x_j}^2} \qquad c = \frac{c_{ij}}{\Delta x_i \Delta x_j} \qquad \text{with} \quad i \neq j .$$

This bound on Δt gives results reasonably close to the genuine stability condition for curvature flow. For a test problem of collapsing spheres (under $F = -\kappa$), instability appeared at a time step between 100% and 110% of the one computed as above. With this method, the overall error generally decreases as the time step is increased, up until the point at which instabilities appear. It is therefore advantageous to take the largest time step that can safely be taken, and the results for the above estimation are satisfactory in that regard.

These equations for Δt (Eq–2.7 in particular) are expensive to compute, however, and would significantly increase the cost of the method if they were computed at every discretization point and at every time step. Fortunately, the restriction on the time step tends to change very slowly, and, given some padding (such as is built into the parabolic stability bound above), the time step usually only requires infrequent checks, once established.

2.6 Initial and Boundary Conditions

Given an initial surface Γ_0 it is necessary to construct a consistent initial condition for the PDE (Eq–1.3) which is to be solved numerically. Specifically, a continuous φ_0 must be chosen to satisfy

$$\{x \in \mathbb{R}^n : \varphi_0(x) = 0\} = \Gamma_0 .$$

This restriction still leaves much freedom in the choice of φ_0, but from a numerical point of view, there is another desirable stipulation, and that is that $\nabla\varphi_0$ not vanish near the zero level set. A good choice of φ_0 is therefore the signed distance function to Γ_0 (negative within U and positive without), which provides $|\nabla\varphi_0| = 1$ everywhere that the gradient is defined.

Also, because of the extension of the analytic problem to all of $\mathbb{R}^n$ coupled with the necessary restriction of the numerical domain to a finite region, an artificial, "far field" boundary condition has to be supplied to the numerical method. A natural condition can be achieved for bounded surfaces by letting φ_0 go to $r - R_0$ for $r \gg R_0$. Here, $r = |\mathbf{x} - \mathbf{x_0}|$, and $\mathbf{x_0}$ and R_0 approximate the center and radius of U. By initializing φ in this fashion, the far field boundary condition can then be taken as the exact solution for the collapsing sphere problem (with $F = a - b\kappa$), as given in §1.4.

2.7 The Successes and Limitations of the Level Set Method

The level set methodology is a very elegant approach to the surface evolution problem, and it has had many successes as a numerical method. These include problems in geometry [4, 9, 12, 36], grid generation [37], fluid mechanics [29, 38, 42], combustion [33], solidification [40], device fabrication [2], and image processing [26, 27].

Its fundamental limitation is an exclusively computational one. When the problem is recast from a surface evolution problem into an evolution in one higher dimension, the

computational workload suffers. If k is the dimension of the surface, and $n = \frac{1}{h}$ with h describing the length scale of the computational resolution, then the cost of tracking the surface can reasonably be expected to be $\mathcal{O}(n^k)$ per time step. Both the marker particle and the volume of fluid methods meet this expectation, but the level set method, as described above, requires $\mathcal{O}(n^{k+1})$ to update the solution. This scaling of the workload places a severe restriction on the degree of complexity that can be effectively considered in a computation.

Recently, two methods have been developed to reduce the cost of advancing the solution from $\mathcal{O}(n^{k+1})$ back to $\mathcal{O}(n^k)$. One of these is the narrow band technique in which the computational domain is restricted to a small distance about the the zero level set [1, 8]. The other method is the one that is developed in this thesis, and that is the numerical solution of the PDE (Eq–1.3) on an adaptive mesh.

Through the use of the adaptive mesh described in the next chapter, it is possible to concentrate the work near the zero level set, and the result is an $\mathcal{O}(n^k)$ algorithm much resembling the narrow-band technique. One distinction between the two lies in the fact that the latter requires a re-initialization of φ each time the narrow band is moved. The adaptive mesh version avoids this issue by supplying coarse resolution data away from the surface which can be used to initialize values as the mesh changes.

The adaptive mesh also provides two other distinct advantages. First, it allows the computational domain to be extended well beyond the sufrace of interest, without the incurrence of a perfomance penalty. This alleviates the concern that the imposed boundary conditions may adversely affect the solution, and allows the use of the "natural" far field boundary condition suggested in the previous section. Secondly, the adaptive mesh also allows for a non-uniform resolution of the surface itself. The ability to selectively redistribute the density of information (and correspondingly, the workload) accross the surface gives the adaptive method a unique advantage in matching itself to the small scale features of the surface. This, together with the avoidance of re-initialization, are the primary features distinguishing it from the narrow-band methods.

Acurately computing the solution on an adaptive mesh proves to be a non-trivial task, however — and particularly so in the case of parabolic, curvature dependent motion. The bulk of the work in this thesis will, in fact, be directed toward the solution of parabolic type equations on an adaptive mesh.

Chapter 3

Adaptive Mesh Refinement

This chapter describes the adaptive mesh that will be taking the place of the uniform grid that was assumed in the last chapter. It also covers a collection of generic algorithms associated with the mesh and its maintenance, but leaves for the following chapter the crucial question of how the schemes for updating the solution on a uniform grid (as in §2.3 and §2.4) will be extended to update a solution on the adaptive mesh.

3.1 Definitions

A cell, C, is simply a rectangular volume (parallel to the coordinate axes) with center $\mathbf{X}(C)$ and dimensions $\Delta x_C \times \Delta y_C \times \Delta z_C$. (Values of functions such as φ kept at cell centers are generally given as $\varphi(C)$). The adaptive mesh will be constructed as a time dependent set of such cells. The domain, $\mathbb{D}$, of the mesh (the "computational domain") is a fixed rectangular volume with $\cup C = \mathbb{D}$. There is a subset of $\mathcal{M}$ of non-overlapping cells of a uniform size, $\Delta x_0 \times \Delta y_0 \times \Delta z_0$ which span the domain; this is the "coarse grid" upon which $\mathcal{M}$ is built. There is also an integer, $rr > 1$, "the refinement ratio," that describes how the mesh can be refined, in a hierarchical fashion. Any cell, $C \in \mathcal{M}$, can be subdivided into the rr^3 cells of size $\frac{\Delta x}{rr} \times \frac{\Delta y}{rr} \times \frac{\Delta z}{rr}$ which span C. These cells are known as the "children" of C, and are given by $\mathbf{Child}(C)$, and C is called their parent, $\mathbf{Parent}(ch)$. Properly, operators such as **Child** and **Parent** are maps amongst the subsets of $\mathcal{M}$ — including the empty set (many cells have no children, and members of the coarse grid have no parents). It is, however, convenient to consider them as maps on $\mathcal{M}$ when the subsets contains one element and allow notations such as $\mathbf{X}(\mathbf{Parent}(C)) - \mathbf{X}(C)$, which makes perfect sense so long as

$\mathbf{Parent}(C) \neq \emptyset$.

The above hierarchy defines the basic structure of all meshes — the level of formality may seem on the exorbitant side, but it will prove to make later definitions and constructions clearer and simpler (even up to the level of implementation into code). Even at this point, the formal structure of the mesh can be examined to codify its important features and to provide further definitions that will be useful.

From the recursive nature of the cells in $\mathcal{M}$ it follows that every cell must have a size given by

$$\Delta x_i \times \Delta y_i \times \Delta z_i \;=\; rr^{-i}\Delta x_0 \times rr^{-i}\Delta y_0 \times rr^{-i}\Delta z_0$$

for some integer $i \geq 0$. The cells can be grouped accordingly into $L_0, L_1, \ldots, L_f$. (The mesh is assumed to be finite and therefore have a smallest cell whose size is given as above with $i = f$.) The L_i's are called the "levels" of $\mathcal{M}$, and the following relationships hold between them:

$$\mathbf{Child}(L_i) = \begin{cases} L_{i+1} & 0 \leq i < f \\ \emptyset & i = f \end{cases} \qquad \mathbf{Parent}(L_i) = \begin{cases} \emptyset & i = 0 \\ L_{i-1} & 0 < i \leq f \end{cases}$$

L_0 is called the "coarsest level" and L_f the "finest level." $\mathbf{Lev}(C)$ maps C to the level of which it is a member. A pictorial representation of this grouping is given in Fig–3.2, which should serve to clarify the structure. A level $L_i \subseteq \mathcal{M}$ is also, of course, a subset of a uniform grid corresponding to a fully refined level of $\mathcal{M}$ which will be denoted as $\overline{L_i}$.

Operators such as the shift operators on $\overline{L_i}$ can then be applied on L_i as $\mathbf{S_d}(C)$ for $\mathbf{d} \in \{\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}, -\hat{\mathbf{x}}, -\hat{\mathbf{y}}, -\hat{\mathbf{z}}\}$, with an additional allowance for $\mathbf{S_d}(C) = \emptyset$. Using the shift operators, a measure of the way in which a mesh is refined can be established which reflects the degree to which the levels are "nested" within their parents. The nesting of a cell can be measured as the minimum number of shifts that have to be applied to its parent, P, to produce the empty set: $\mathbf{S}_{\mathbf{d}_n} \circ \ldots \circ \mathbf{S}_{\mathbf{d}_1}(P) = \emptyset$. The mesh, $\mathcal{M}$ is then said to be n-nested if the minimal nesting of all cells in $\mathcal{M} \backslash L_0$ is n.

The formal structure of $\mathcal{M}$ will also become the basis for the data structure in the implementation of the numerical methods. An overview of that data structure appears in Fig–3.1. The primary structure is that of a tree, which corresponds to the hierarchical nature of $\mathcal{M}$, with pointers to parents and children, but overlying this are lists linking together the cells of each level and links to the neighbors given by the shift operators. The links to the neighbors are required for easy movement about the levels (e.g. for evaluating finite differences) since indexing cannot be used in the absence of a uniform grid.

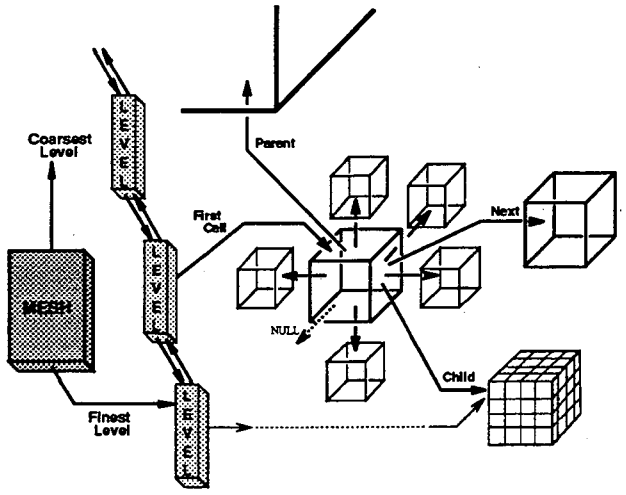


Figure 3.1: The Data Structure for the Mesh

One other convenient way of representing a mesh is through the set of its childless cells. They cover $\mathbb{D}$ and are accordingly called the "finest covering of $\mathbb{D}$." From them the rest of $\mathcal{M}$ is fully determined, so they do, in fact, provide a representation of $\mathcal{M}$, and one which is easily portrayed. An example appears in Fig–3.3, which actually shows the finest covering of a cross-section of the computational domain.

3.2 Maintaining the Mesh

The generic algorithm used to build the initial adaptive mesh and to maintain it throughout the computation is given below. Here, the refinement is checked at regular intervals of t_a, but more elaborate schemes for determining when it should be checked can easily be incorporated.

1. Create L_0 and initialize the computational variables on this grid.
2. $k = 0$; Flag cells needing refinement.
3. If any cells have been flagged:
   (a) Create children under each flagged cell.
   (b) Fixup the data structure.
   (c) Initialize variables in each new cell.

(d) Return to step 2.

4. Advance the solution on $\mathcal{M}$ until $t_{\mathcal{M}} = min(kt_a, t_f)$.

5. $k = k + 1$ Flag cells that need to be coarsened or refined.

6. If any cells have been flagged:

   (a) Remove all descendents under every C-flagged cell.

   (b) Create children under each R-flagged cell.

   (c) Fixup the data structure.

   (d) Interpolate variables into each new cell.

7. If $t_{\mathcal{M}} < t_f$, return to step 4

At each check, the mesh can be adjusted by the incremental addition of children or through the removal of all descendents of certain cells. In the initial construction of the mesh, repeated checks are made in order to allow the mesh to grow to arbitrary depths of refinement. No coarsening is allowed in this loop to insure its termination (assuming a maximum depth of refinemet). A single coarsening is allowed after this loop to accommodate the removal of the excess. The later checks only allow a single increase in the local refinement per occurrence.

In each case, the maintenance is carried out in two steps; first cells are flagged for refinement and then the mesh is modified based on these flags. This encapsulation of the machinery needed to update the data structure allows for the easy implementation of arbitrary refinement techniques. The crucial steps, from an implementation perspective, are 3b and 6c during which the pointers in cells affected by the adaptation have to be updated without significant computational overhead. This can be accomplished through the use of incremental updates in which only the relevant neighboring cells in the data structure have their pointers corrected. Under that paradigm, the maintenance of the mesh becomes transparent in terms of computational cost.

The initialization in step 3c assumes that the initial data is available at arbitrary locations. Initialization of data for cells created later in the computation is carried out by interpolation (typically second order) in step 6d.

3.3 Basic Refinement Techniques

In application to the level set equation for tracking the motion of a surface, the primary use of the adaptive mesh will be to concentrate the data points near the surface of interest — i.e. near the zero level set of φ. In Fig–3.2 there is a pictorial representation of the resulting style of mesh. If the width (in cells) of the refined band on each level is $\mathcal{O}(1)$, and the work done per cell per time step is also $\mathcal{O}(1)$ then the total work per time step will be $\mathcal{O}(n^2)$ for this type of mesh. Such a scaling of the work is what is ideally desired in a front tracking scheme (and is also provided by the marker particle and volume of fluid methods of §2.1). The $\mathcal{O}(n^3)$ scaling of the traditional level set method can become too expensive in many problems, and the use of an adaptive mesh provides one way of combating it — in this way, it is much like the narrow band methods of [1].

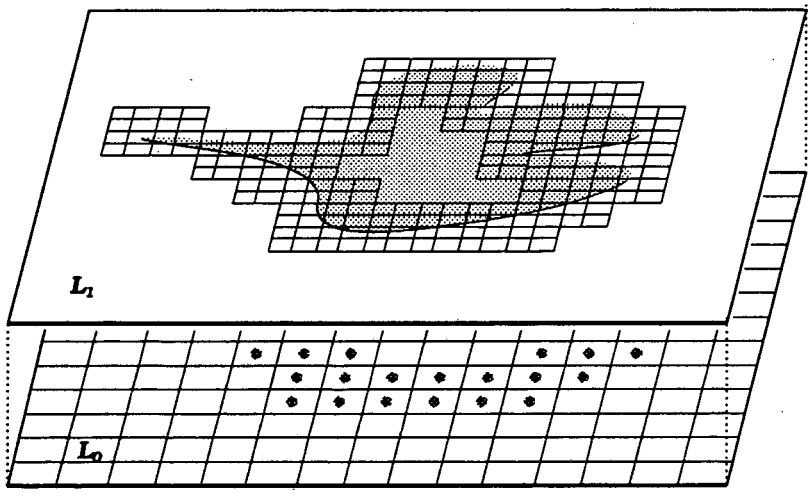


Figure 3.2: The Banded Mesh Structure

But an adaptive mesh also offers an additional advantage, and that is the allowance of increased resolution in particular regions of the level surface. For example, in the collapsing dumbbell problem illustrated in Fig–3.3, an extra level of refinement has been added along the neck to better resolve the rapid motion and impending topological change there. And this advantage is not limited to such artificially constructed problems. In many physical problems (with a curvature dependent motion) regions of very high relative curvature appear which could be adequately resolved on an adaptive mesh. The allowance of this style of refinement will become a critical issue in the development of numerical techniques

for the level set equation on the adaptive mesh.

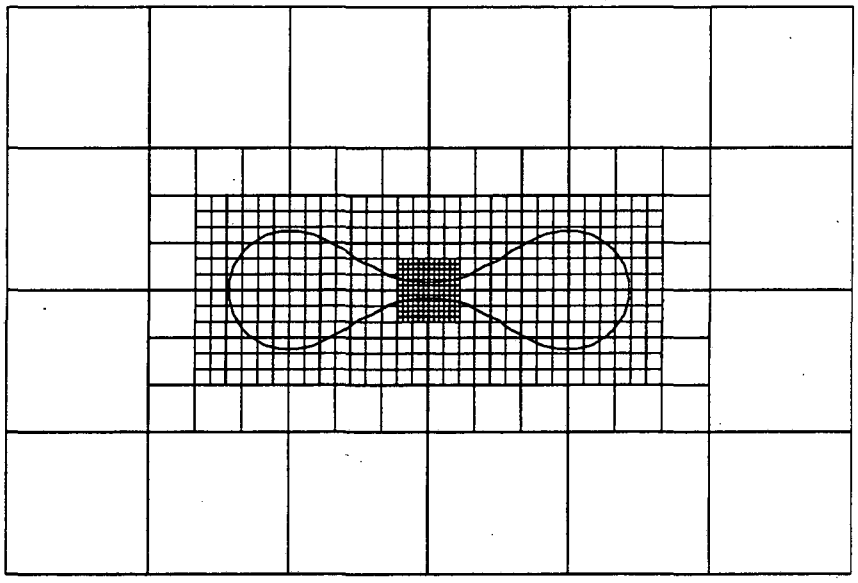


Figure 3.3: Classic Dumbbell Problem under Mesh Refinement

3.4 Extending Numerical Methods

The first step is to outline the way in which numerical methods for uniform grids can be extended for application on an adaptive mesh. The extensions will be applied to the levels of $\mathcal{M}$ since these locally look like uniform grids.

Let $\mathbf{A}_{\Delta t}$ be a numerical method for updating (by Δt) a solution φ on the uniform grid $\overline{L}$ corresponding as before to the "fully refined" version of L:

$$\varphi_{\overline{L}} \leftarrow \mathbf{A}(\varphi_{\overline{L}}) .$$

Now φ is only defined on L, but it can be represented on $\overline{L}$ as a map to $\mathbb{R}^*$ with the symbol ∞ serving as a place holder for the points at which it is undefined:

$$\varphi^* = \begin{cases} \varphi(C) & C \in L \\ \infty & C \notin L \end{cases} .$$

Then, using the convention that $\mathbf{A}(\varphi^*)$ evaluates to ∞ wherever it has a non-zero dependence on cell holding an ∞, the method can be partially applied to L through the operator:

$$\mathbf{A^L} : (\mathbb{R}^*)^L \rightarrow (\mathbb{R}^*)^L \qquad \mathbf{A^L}(\varphi) \equiv \mathbf{A}(\varphi^*_{\overline{L}}) .$$

If it is based on a uniform grid method that only uses nearby values to update the solution in a given cell (as is the case for the methods discussed in §2.3 and §2.4), the application of the above formulation would result in the encroachment of the undefined values only along the "boundary" of L as a subset $\overline{L}$. The cells of L which receive a value of ∞ in a single application of $\mathbf{A}^{\mathbf{L}}$ are called the "periphery" of L (with respect to the method); the remaining cells are called the "interior" of L.

All that is needed, then, to extend the numerical method to the adaptive mesh, is means for constructing the new values of φ along the periphery of L using the known values of φ on $\mathcal{M}$. Chapter 4 will introduce the way in which this can be accomplished for the schemes of §2.3 and §2.4, but the generic algorithm can be given now for advancing the solution on a fixed mesh $\mathcal{M}$ to a time t. Assuming for the moment the existence of the peripheral update, the following recursive procedure advances the solution on a level, L_i, to the time t. This procedure will then advance all of $\mathcal{M}$ when called on its coarsest level (i.e. with $L_i = L_0$).

Advance(L_i,t):

1. Compute a timestep, Δt, for L_i.
2. Apply $\varphi_{L_i} \leftarrow \mathbf{A}^{\mathbf{L}}{}_{\Delta t}(\varphi_{L_i})$.
3. Update the periphery of L_i using $\varphi_{L_{j\leq i}}$.
4. If $L_{i+1} \neq \emptyset$, then **Advance**(L_{i+1},t_{L_i}).
5. If $t_{L_i} < t$, then return to Step 1.
6. If $i \neq 0$, then correct φ_{L_i-1} with φ_{L_i}.

In step 6, the values in the parents of a level are overwritten with the newly computed values in the fine cells underneath them, effectively coupling the solution on the coarser level to that on the finer levels. For odd refinement ratios, the coarse cell will share its center with a fine cell and the value there can simply be copied up. For even refinement ratios, interpolation from the fine cells surrounding its center is required (a second order scheme is used).

The procedure given above also makes the computation fully adaptive in time; i.e. each level is allowed to choose its on time step and computed values on the mesh will therefore be staggered in time. Coarse levels will take a large time step, and then the finer

levels will take many small time steps to cross the same interval and eventually pass the time reached by their parent (on their final step). Consequently, the overwrite discussed above will typically involve an additional linear interpolation in time. As mentioned in §2.5 it is often beneficial to take a large time step, so this temporal adaptivity can improve performance as well as efficiency. The appropriate time step can be computed as it would for the uniform method in the interior of L_i; further considerations may also have to be included to account for restrictions imposed by the peripheral update.

3.5 Boundary Conditions

The numerical schemes of §2.3 and §2.4 also require the incorporation of boundary conditions for the PDE as discussed in §2.6. These can be implemented in two basic ways; by modifying the dependence of the scheme on its stencils at the boundaries or by including artificial "boundary" cells which mimic the appropriate behavior. On an adaptive mesh, the latter will prove much simpler to implement owing to the large number of possible stencils resulting from the various patterns of refinement that can occur at the boundary.

A swath of cells is maintained along the exterior of $\mathbb{D}$ whose refinement pattern reflects the pattern across the boundary (and inside of $\mathbb{D}$). The far field boundary condition based on the solution of the propagating spheres problem (of §1.4) can then easily be imposed by supplying these boundary cells with the appropriate values computed from the exact solution. A Neuman condition can also be implemented by having each boundary cell in the swath copy the value from its "image" cell across the boundary of $\mathbb{D}$. (A similar construction can be used for periodic boundary conditions.)

3.6 Interpolation and Triangulation

Standard routines on a uniform grid usually become significantly more complicated in translation to an adaptive grid. A good example is that of interpolation. In the previously mentioned special case of interpolation from children to an overlying parent cell, it is clear how to proceed since the target point is effectively within a uniform mesh. A general means of producing an interpolated value at an arbitrary position in the domain is less straight forward. Given φ at the centers of the cells of the finest covering, $\mathcal{F} \subset \mathcal{M}$, the following procedure produces a continuous first order approximation to φ across the domain.

The basic idea is to first interpolate values at the corners of the cells in $\mathcal{F}$ and then interpolate across their interiors with bilinear interpolation. The trick is in interpolating the corner values in such a way the continuity is assured. To that end, the corners are divided into two sets; those which are bordered by eight cells and those which are not (the latter lie at the transitions between levels and are what distinguish the adaptive mesh from a uniform one). For the former set, values are obtained immediately by an average, weighted by distance, of the values at the eight bordering cells. If ${L^*}_c$ is the coarsest level in the covering, ${L^*}_c = L_c \cap \mathcal{F}$, then corners of all of its cells must be in this set. Any corners from the finer levels that lie on the faces (and edges) of its cells — these are in the latter set — can then be forced to match the bilinear interpolation from the corners across the faces. After that step, all corner values for the cells in ${L^*}_{c+1}$ will have been obtained, and the procedure can be continued recursively through the levels.

For the level set equation, an important consequence is the ability to produce a first order triangulation of the level surfaces of φ. Given the above interpolation of φ, a marching cubes style algorithm [25] can be immediately applied on the finest covering. The continuity of the interpolation then guarantees that this triangulation is closed.

3.7 Judging Performance

Lacking an analytic proof of performance, the best way to judge the performance of an algorithm on the adaptive mesh is to compare its solutions to the exact solutions on a set of test problems (i.e. problems for which the exact solutions can be written down). The choice of both the test problems and the means of comparison are critical. Choosing the former is, necessarily, application specific and something of an art, but the latter can be addressed more generally and concretely here.

Let φ_c denote the computed solution and φ_e the exact solution. The standard global performance estimates such as $||\varphi_c - \varphi_e||_2$ easily mask important information when applied on an adaptive mesh. If the mesh is refined in a region of high error, for example, then a method on it could easily achieve an overall reduction in error even while performing badly along the periphery of the levels (the increase in error there being outweighed by the reduction in the interior). On the other hand, the mesh may be left coarse in some region because the error there is of no importance, and it may be beneficial to sacrifice the accuracy there while striving for high accuracy in another region. Such would be the case

with the level set equation when the only accuracy of any concern is that at the zero level surface. But in a norm estimate, the error in the refined zone would be swamped by the surrounding (unimportant) error.

For the level set equation, the specific fix of giving the error only as it appears on the zero level surface could be made. It could be given as the volume between the computed and exact surfaces, a plot of the distance between them, or another norm based on that distance. Unfortunately, the computed surface must first be reconstructed from φ, and if the method of reconstruction is not sufficiently accurate, the error it induces will mask the computational error it was intended to measure.

Having the exact solution φ_e everywhere allows for an easy solution, which is to plot $|\varphi_c - \varphi_e|$ across the finest covering of the domain. Such plots prove to be very useful in the analysis of the adaptive mesh schemes, especially in highlighting the onset of difficulties, be they in a "region of interest" or not. This will be the tact taken in the analysis of the computations to follow. Typically, however, these computations will have been conducted in three dimensions so the data will first be sliced on a plane perpendicular to one of the coordinate axes (see Fig–4.2). The domain of the plot is taken as a rectangular subset of this plane. The error, $|\varphi_c - \varphi_e|$, is then evaluated at the cell centers of the finest covering of this domain (which will be only piecewise planar, the cell centers being shifted out of the slicing plane) and plotted.

Chapter 4

The Level Set Method Under Adaptive Mesh Refinement

To formulate a method for updating the solution on a level L_i of $\mathcal{M}$, the numerical techniques given in §2.3 and §2.4, can be applied in the fashion just described to update the solution in the interior cells of L_i. The crucial step is then to provide a means for updating the peripheral cells where the first update was undefined. A criterion also needs to be established that can be used to determine whether a given scheme for updating the periphery is a good one or not. This can be supplied by the fundamental requirement that the addition of cells in the adaptive mesh should never make the solution worse (even though they may make the solution no better if added arbitrarily). The immediate corollary is that the solution on an adaptive mesh should never be worse than the solution on the underlying coarse grid alone.

As was the case in the development of the uniform grid methods, the hyperbolic and parabolic terms will require separate and specialized treatments. The hyperbolic term will turn out to be the much simpler of the two, so it is dealt with first.

4.1 The Hyperbolic Term

The most straight forward way of obtaining the new values on the periphery of L_i is by interpolating them from the known values on $L_{j\leq i}$. There are a variety of ways in which such an interpolation can be carried out, ranging from simple injection of the value from the overlying coarse cell (the parent), to linear interpolation off of a cube of

surrounding coarse cell centers, to a higher order method. Nearby interior cells of L_i could also be included in the interpolative scheme since they will have been successfully updated before the interpolation occurs.

For the upwinding schemes of §2.3 applying such an interpolation along the periphery does not degrade the numerical solution and is therefore a fine way to patch the numerical scheme for a uniform grid onto the adaptive mesh. In fact, linear interpolation off of the surrounding coarse data points (with linear interpolation in time as well) is found to be sufficient. Since the hyperbolic term is not necessarily the only contributor to φ_t, it is actually the hyperbolic contribution to φ_t that is interpolated from the coarse data, and the time integration of this value then corresponds to the linear interpolation of φ in time.

It should not come as a surprise that interpolation on the periphery works well for the hyperbolic term. Indeed, such interpolative schemes have been shown to work well by Berger, Colella and Oliger for a variety of hyperbolic problems [6, 7]. In their work, certain quantities typically have to be conserved, and in order to assure their conservation, a correction has to be applied following the interpolation. In the case of the hyperbolic term of the level set equation, no such correction is required.

As a foreshadowing of what will prove important when the parabolic term is considered, two basic tests are performed. Both are for an initial function with spherical level surfaces centered about the origin: $\varphi = r - 1$. In the first test a level of refinement is maintained about the zero level surface, and in the second the refined zone is restricted to a patch near the positive z-axis. The patch is given by the intersection of the band in the first problem with a cone of angle $\frac{\pi}{6}$ about the axis. Such a patch is pictured in Fig–4.1 along with a cut-away level surface; the cells that are shown are actually the coarse cells which have children. Fig–4.2 shows the typical way in which the data is sliced to allow the error to be plotted. Finest coverings of such a slice for the two test problems are shown in Fig–4.3.

Also in these tests, the adaptive time stepping has been suppressed, and the solution on all the levels has been advanced in lockstep, the appropriate time step being enforced by the requirement on the finest level. In §4.8 the issue of adaptive time stepping will be returned to and investigated, but for now the analysis can be simplified if the schemes are first considered in a lockstep implementation.

The performance of the method with a band of refinement is presented in Fig–4.4 where the error from a central slice is plotted at $t = 0.3$ (the plot has also been restricted

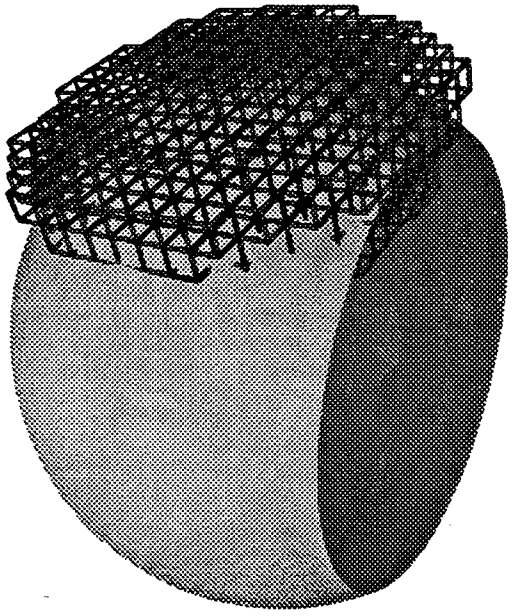


Figure 4.1: Patch of Refinement on Level Surface

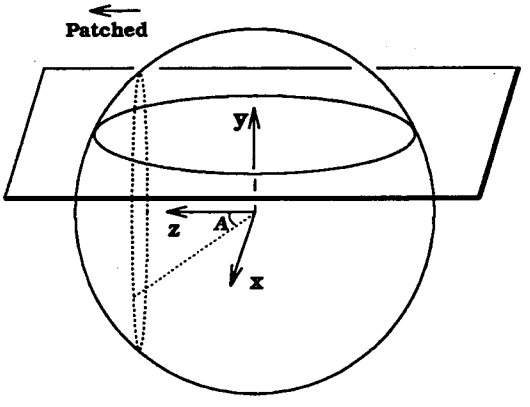
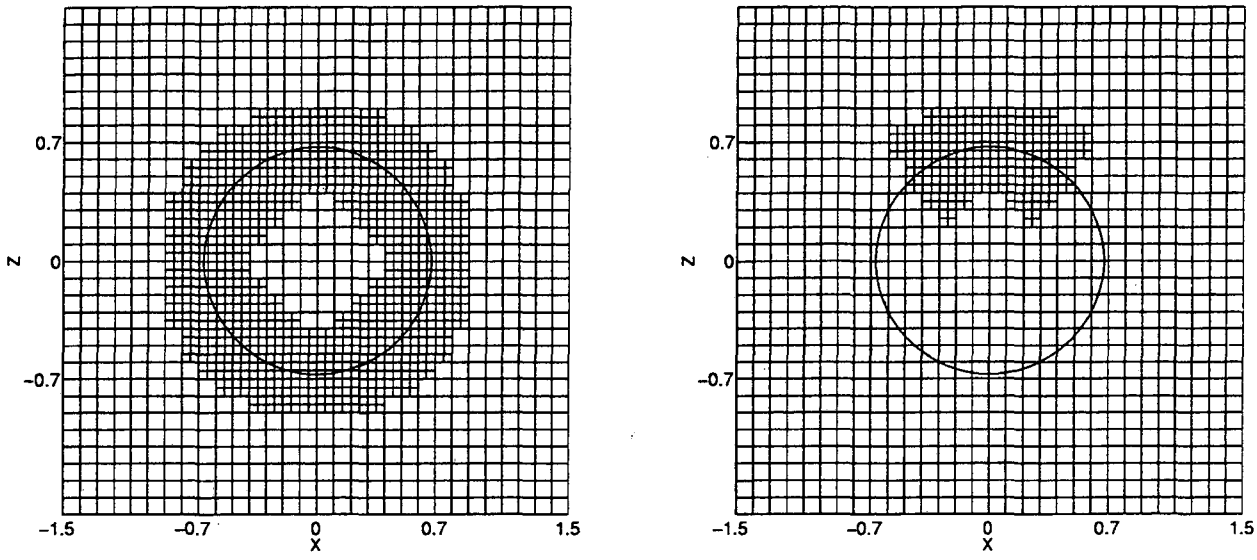


Figure 4.2: Slicing of 3D Computation for Error Plot



(a) Refinement in Band

(b) Refinement in Patch

Figure 4.3: Slices from the Two Styles of Test Problems

to the center of the slice — the domain extends farther). The data from the fine cells is plotted in black and that from the coarse cells in grey, and the error is shown on a log scale to emphasize its reduction by an order of magnitude within the refined band. The method in fact looses very little accuracy in comparison to a full method run at the fine grid resolution. The increase in error over the full method is shown in a portion of the refined band in Fig–4.5, here it can be seen that the solution in the center of the band (where the zero level set is) attains an accuracy almost equal to that of the full method.

For the test in which the refinement is limited to a patch, the corresponding error and comparison plots are given at $t = 0.3$ in Fig–4.6 and Fig–4.7. Again, the error within the patch in near that of the full method. In this case, the numerical diffusion in the underlying method results in the increase in error seen along either edge of the patch.

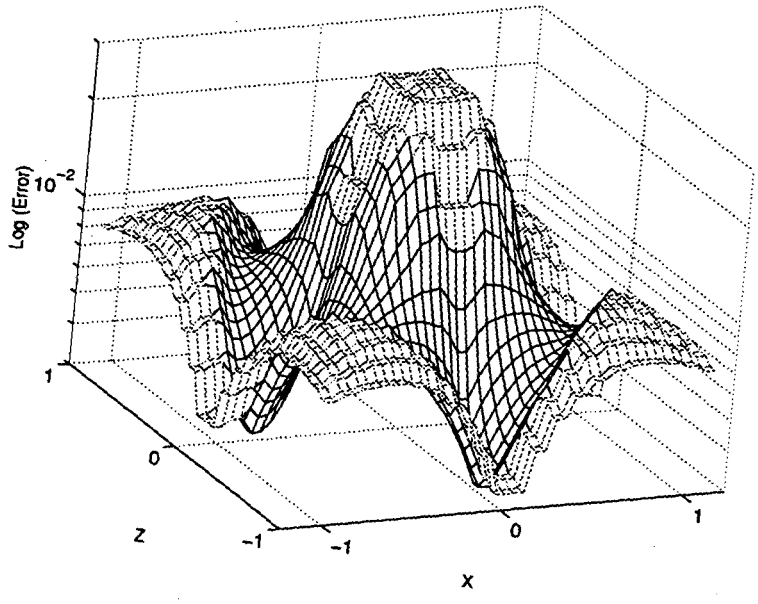


Figure 4.4: Error for Banded Refinement

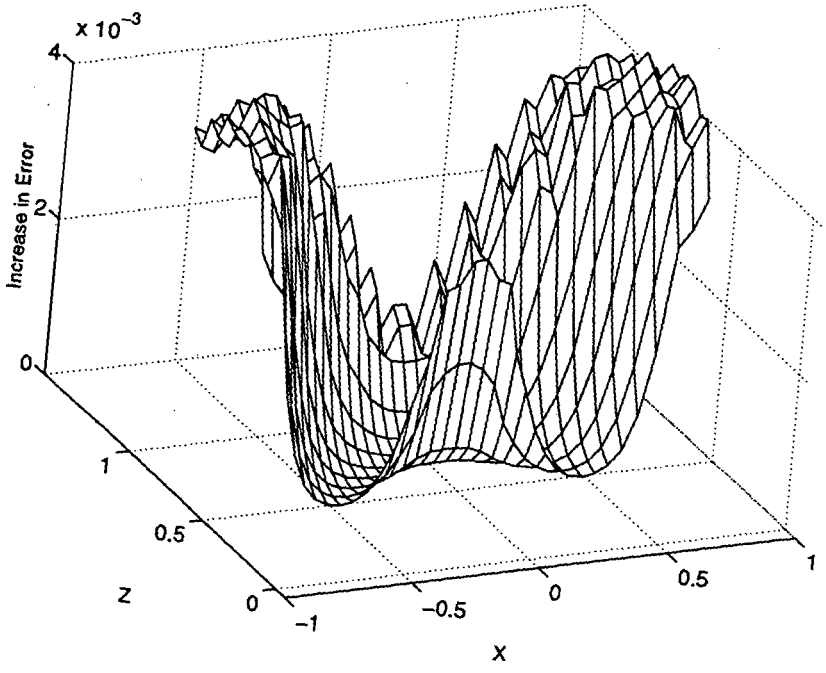


Figure 4.5: Comparison of Banded Refinement to Full Method

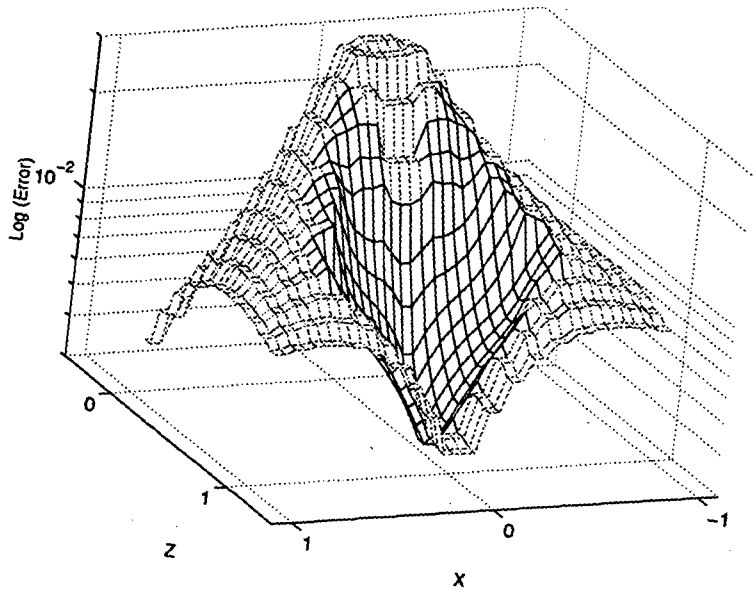


Figure 4.6: Error for Patched Refinement

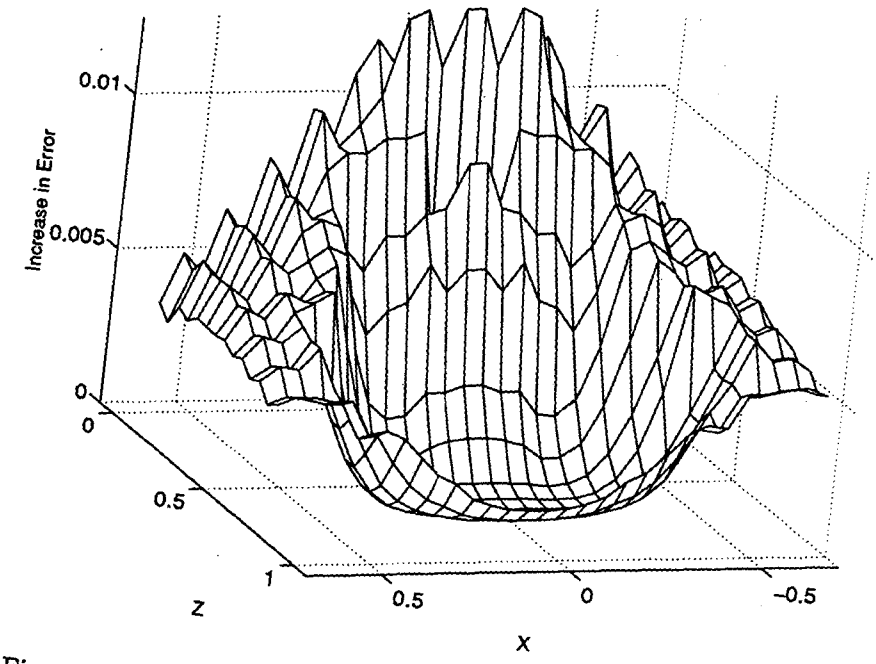


Figure 4.7: Comparison of Patched Refinement to Full Method

4.2 The Parabolic Term

For the parabolic term, using interpolation (of any style and order) along the periphery will lead to unsatisfactory results. The difficulty is illustrated in Fig–4.8 where a refined patch similar to that of the previous section has been applied to a sphere collapsing under curvature. Along the edges of the patch, the error in the fine cells exhibits oscillations which grow in time, with the solution eventually becoming corrupted on the coarse grid as well.

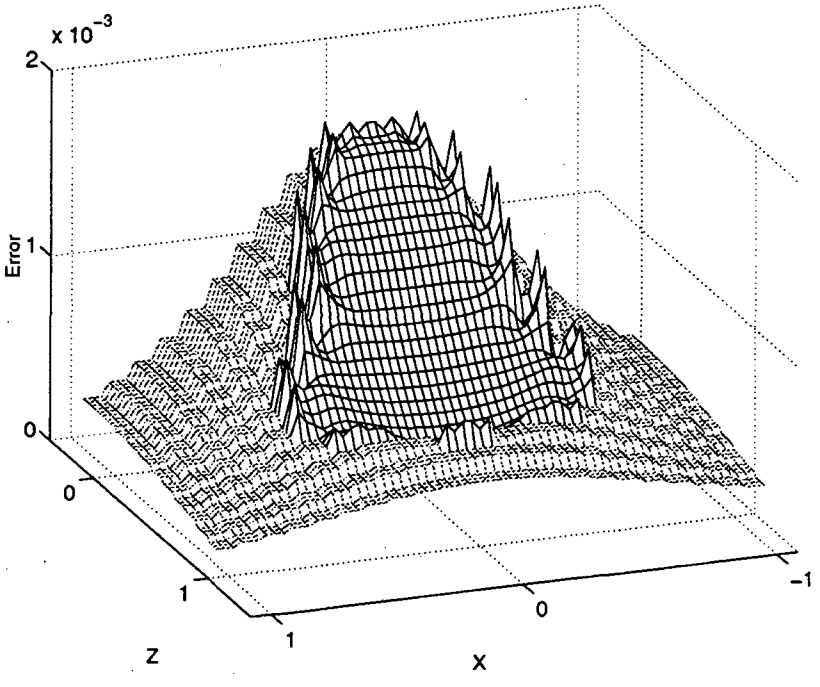
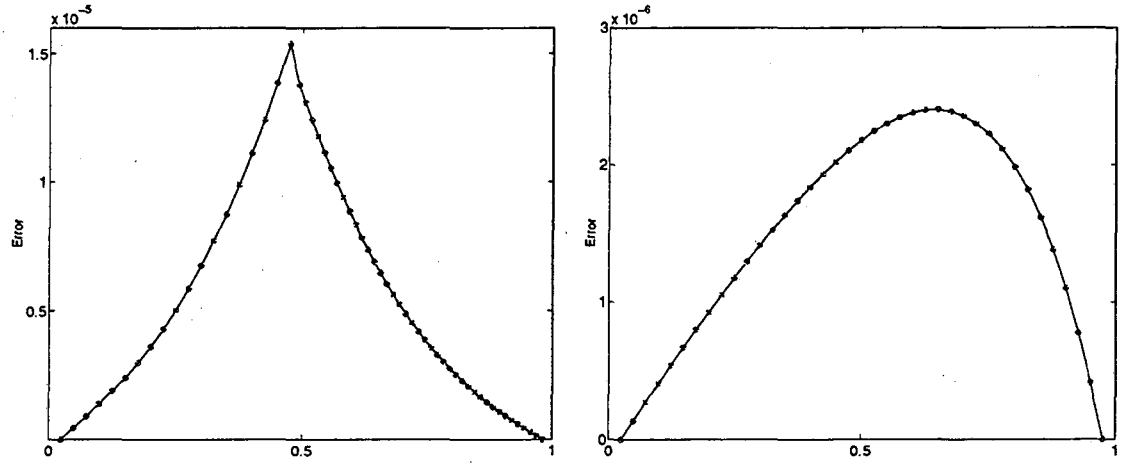


Figure 4.8: Failure of Interpolation for Curvature Flow

It turns out that interpolation is, in general, a bad idea in solving parabolic type equations on an adaptive mesh. The next few sections examine the difficulty and means to its resolution in the context of two prototypical parabolic equations: the nonlinear equation for curvature flow and the linear heat equation. The difficulty can be seen in an attempted solution of the one-dimensional heat equation which is therefore the natural place to attempt to understand what has gone wrong and how it can be fixed.

4.3 The One-Dimensional Heat Equation

This section considers as a test problem the solution of the one-dimensional heat equation, $u_t = u_{xx}$, on an adaptive mesh in the unit interval. The mesh is taken to consist of a uniform coarse grid of cells on $[0,1]$ with each of the cells in $[\frac{1}{2},1]$ subdivided into two fine cells. Thus the only peripheral cell is the leftmost fine cell, and the value there can be interpolated from the coarse cell lying immediately to its left and the fine cell to its right. Elsewhere, the standard center-difference formulation can be used. The computation (and the ones to follow) will be carried out with twenty coarse cells and an initial condition of a sine wave, $u_0 = \sin x$, with the exact solution therefore being $u_e(x,t) = e^{-t}\sin x$. Under the interpolative scheme, the computed solution at $t=0.05$ (Fig–4.9) is an order of magnitude worse than the solution on the course grid alone, which is shown in the right hand plot. In these plots the data points (the cell centers) are indicated by circles.



(a) Adaptive Mesh (b) Uniform Coarse Mesh

Figure 4.9: Failure of Interpolative Method for the Heat Equation

The error plotted in Fig–4.9 gives a hint as to what has gone wrong. The jump in u_x at the interface between the coarse and fine grids is indicative of the presence of a numerical source term there. Fig–4.10 shows the growth of the error in time as the contribution from this source accumulates. Furthermore, if the spatial resolution is doubled (that is, if each cell is halved), then there is a corresponding reduction in the error, just as the extent of the source is cut in half. The problem here resides in the fact that conservation (of heat) is not being maintained at the peripheral cell. It is not that the scheme is unstable — indeed,

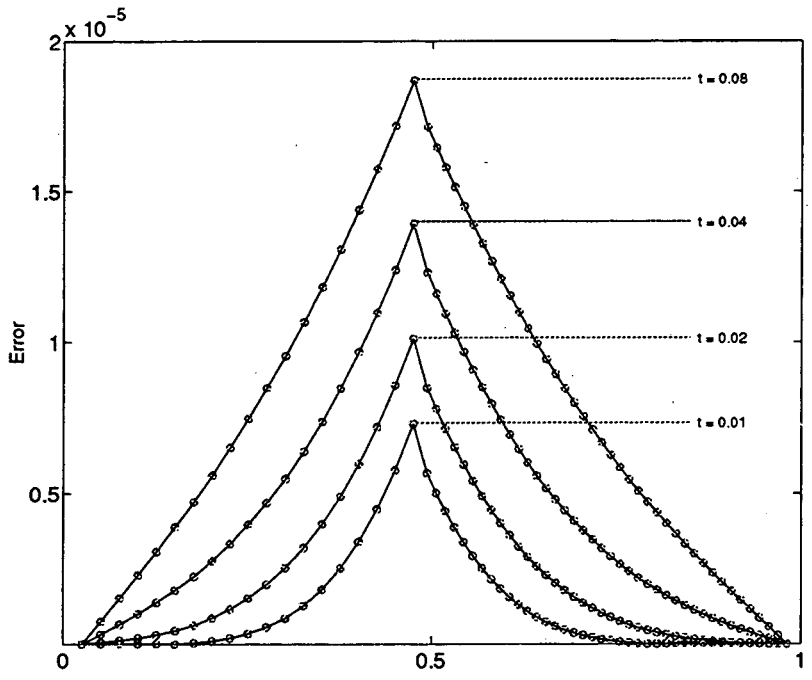


Figure 4.10: Accumulation of Source Term

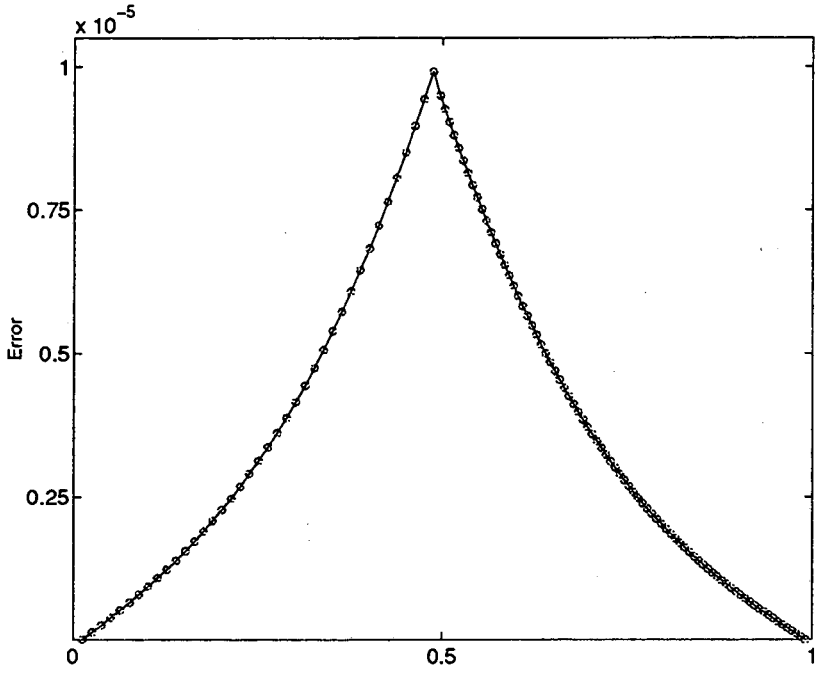


Figure 4.11: Reduction of Source Term under Refinement

Berger has shown such constructions to be stable in [5] — or that it does not converge to the correct solution, but just that it does so in a very inaccurate fashion.

At this point, there are several ways to proceed toward a resolution of the problem. An artificial source could be placed at the grid interface to counter-balance the numerical source, or a flux balance could be enforced there, or a means of computing u_{xx} at the leftmost fine cell could be devised. Each of these paths will lead to the same answer, but the last will also eventually show the way to a method for the generalized motion by mean curvature.

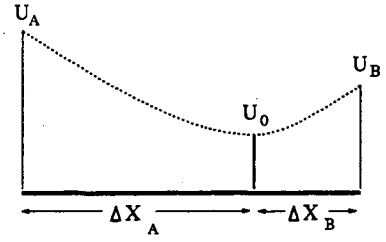


Figure 4.12: Construction of Spline at the Periphery

Let 0 denote the leftmost fine data point, A its coarse neighbor to the left, and B its fine neighbor to the right. Write their values as U_0, U_A, and U_B, and their relative positions as 0, ΔX_A, and ΔX_B; see Fig–4.12. Then a quadratic spline

$$U = a + b\Delta X + c(\Delta X)^2 \tag{4.1}$$

can be fit to the data by requiring:

$$\begin{aligned} U_0 &= a \\ U_A &= a - b\Delta X_A + c(\Delta X_A)^2 \\ U_B &= a + b\Delta X_B + c(\Delta X_B)^2 \end{aligned}$$

which yields

$$c = \frac{\Delta X_A(U_B - U_0) - \Delta X_B(U_0 - U_B)}{\Delta X_A \Delta X_B(\Delta X_A + \Delta X_B)},$$

and as formal differentiation of Eq–4.1 gives $U_{xx} \approx 2c$, a method for updating the leftmost fine point follows:

$$D_t U_0 = 2\frac{\Delta X_A(U_B - U_0) - \Delta X_B(U_0 - U_A)}{\Delta X_A \Delta X_B(\Delta X_A + \Delta X_B)}. \tag{4.2}$$

This method can also be written (or derived) as a balance of flux:

$$\frac{1}{2}(\Delta X_A + \Delta X_B) D_t U_0 = \frac{(U_B - U_0)}{\Delta X_B} - \frac{(U_0 - U_A)}{\Delta X_A} .$$

Application of this method produces errors comparable with those on the coarse grid at the transition in refinement, although for this particular problem there is still a slight increase. The error is plotted in Fig–4.13 along with the error on the coarse grid alone at $t = 0.05$. The behavior at the peripheral cell, while less than ideal, provides at least a starting point for the development of a higher dimensional method.

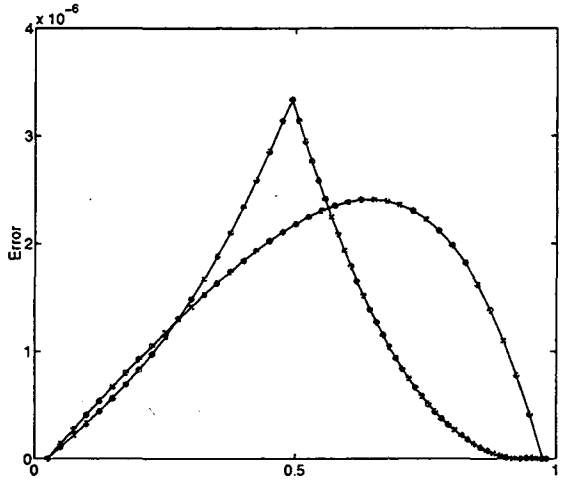


Figure 4.13: Splined and Coarse Grid Solutions

4.4 A Model Problem in Two Dimensions

The next step is to try a similar technique on a model problem in two dimensions. The unit square $[0,1] \times [0,1]$ is covered with a uniform (coarse) grid of cells, and each cells whose center is in $y < \frac{1}{2}$ is refined into $rr \times rr$ fine cells. The dimensions of the coarse grid will be taken to be 30×30 cells throughout this section unless stated otherwise.

The goal is to produce a numerical solution for both the heat and curvature equations which does not become corrupted along the interface (at $y = \frac{1}{2}$) between the coarse and fine cells. A single method can be used for both evolution equations if it first produces approximations to the derivatives which can then be incorporated into either evolution equation. It was with this connection in mind that the splined formulation was used in the

one-dimensional problem. A flux formulation, while instructive for the heat equation, would prove useless for the curvature equation, for although the two test problems are similar,

$$u_t = \nabla \cdot (\nabla u) \qquad \varphi_t = \nabla \cdot \left(\frac{\nabla \varphi}{|\nabla \varphi|} \right) |\nabla \varphi|$$

they are fundamentally different in that the latter cannot be written in the conservative form, $\varphi_t = \nabla \cdot (\textit{flux})$.

The construction of the spline, however, can easily be extended to two (or more) dimensions by writing the general quadratic equation centered at the relevant fine grid point, which for convenience of notation is taken to be the origin:

$$\varphi = a + b_1 x + b_2 y + c_{11} x^2 + c_{22} y^2 + c_{12} xy \,.$$

Then the coefficients b_i and c_{ij} can be determined from the evaluation of this quadratic at five other appropriate data points (this basic idea has been used elsewhere for adaptive mesh schemes [3]), the value of a being established by φ_0:

$$\begin{pmatrix} x_1 & y_1 & {x_1}^2 & {y_1}^2 & x_1 y_1 \\ x_2 & y_2 & {x_2}^2 & {y_2}^2 & x_2 y_2 \\ x_3 & y_3 & {x_3}^2 & {y_3}^2 & x_3 y_3 \\ x_4 & y_4 & {x_4}^2 & {y_4}^2 & x_4 y_4 \\ x_5 & y_5 & {x_5}^2 & {y_5}^2 & x_5 y_5 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ c_{11} \\ c_{22} \\ c_{12} \end{pmatrix} = \begin{pmatrix} \Delta\varphi_1 \\ \Delta\varphi_2 \\ \Delta\varphi_3 \\ \Delta\varphi_4 \\ \Delta\varphi_5 \end{pmatrix} \tag{4.3}$$

where the x_i's and y_i's give the relative positions of the stencil points and the $\Delta\varphi_i$'s their relative values, $\Delta\varphi_i = \varphi_i - \varphi_0$. It is then clear that an "appropriate" choice of stencil must be one for which the above system has full rank and which produces a spline which accurately approximates φ. Since it is intended as an extension to central finite differences, the spline should be fit to nearby points surrounding the evaluation point. When it is formally differentiated to establish the derivatives of φ, these derivatives are effectively being approximated at the "center" of the stencil, since the quadratic is being forced to fit the values on the stencil exactly. Therefore, discrepancies between the center of the stencil and the location of the fine data point for which it was constructed will lead to unsatisfactory results. This issue will be examined in further detail shortly, but first the method can be applied to the model problem with the choice of stencil shown in Fig–4.14.

Recall that the basic concern, for each evolution equation, is whether there exists an initial condition for which the solution under this method on the adaptive mesh is worse

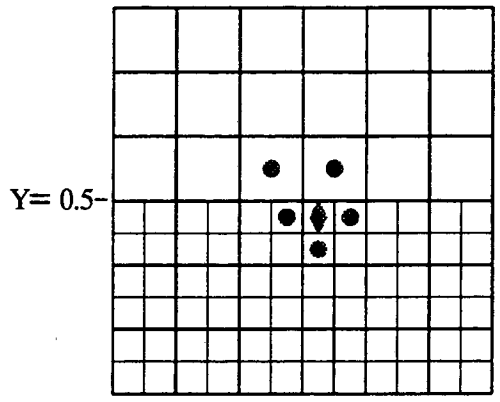


Figure 4.14: Five Point Stencil for the Model Problem

than the solution on the coarse mesh alone. To that end, a useful class of initial conditions for the heat equation are

$$u_0(x) = \sin(k_1 \pi x) \sin(k_2 \pi y),$$

with exact solutions:

$$u_e(x,t) = e^{-\pi^2({k_1}^2+{k_2}^2)t} \sin(k_1 \pi x) \sin(k_2 \pi y).$$

First, for $k_1 = 2$ and $k_2 = 1$, the scheme performs idealy: Fig–4.15 gives plots of the resultant errors, $|u_c - u_e|$, at an early time, $t = 0.001$, and a later time $t = 0.1$. The plots on the left are from the adaptive mesh computation; those on the right are the corresponding coarse mesh computations for comparison. The error initially builds up quickly in the coarse cells, and then diffuses into the fine cells, but even at the later time the error is everywhere reduced from that of the coarse computation. No higher expectation could reasonably be held for this computation.

If, however, the same problem is rotated, with $k_1 = 1$ and $k_2 = 2$, the behavior becomes much worse and the expectation is no longer met. In this case, there is a line of inflection along $y = \frac{1}{2}$, the boundary between the two grids, and along this line difficulties arise. Fig–4.16 shows the same plots as before, taken at the same times. But here, another (large) error immediately appears at the juncture of the grids in a manner that is reminiscent of the artificial source term that is seen for an interpolative scheme. At the later time depicted, $t = 0.1$, the initial sine wave has decayed by two orders of magnitude (and, presumably, the artificial source along with it). The effect of the source is therefore diffused at this point, but it is still apparent.

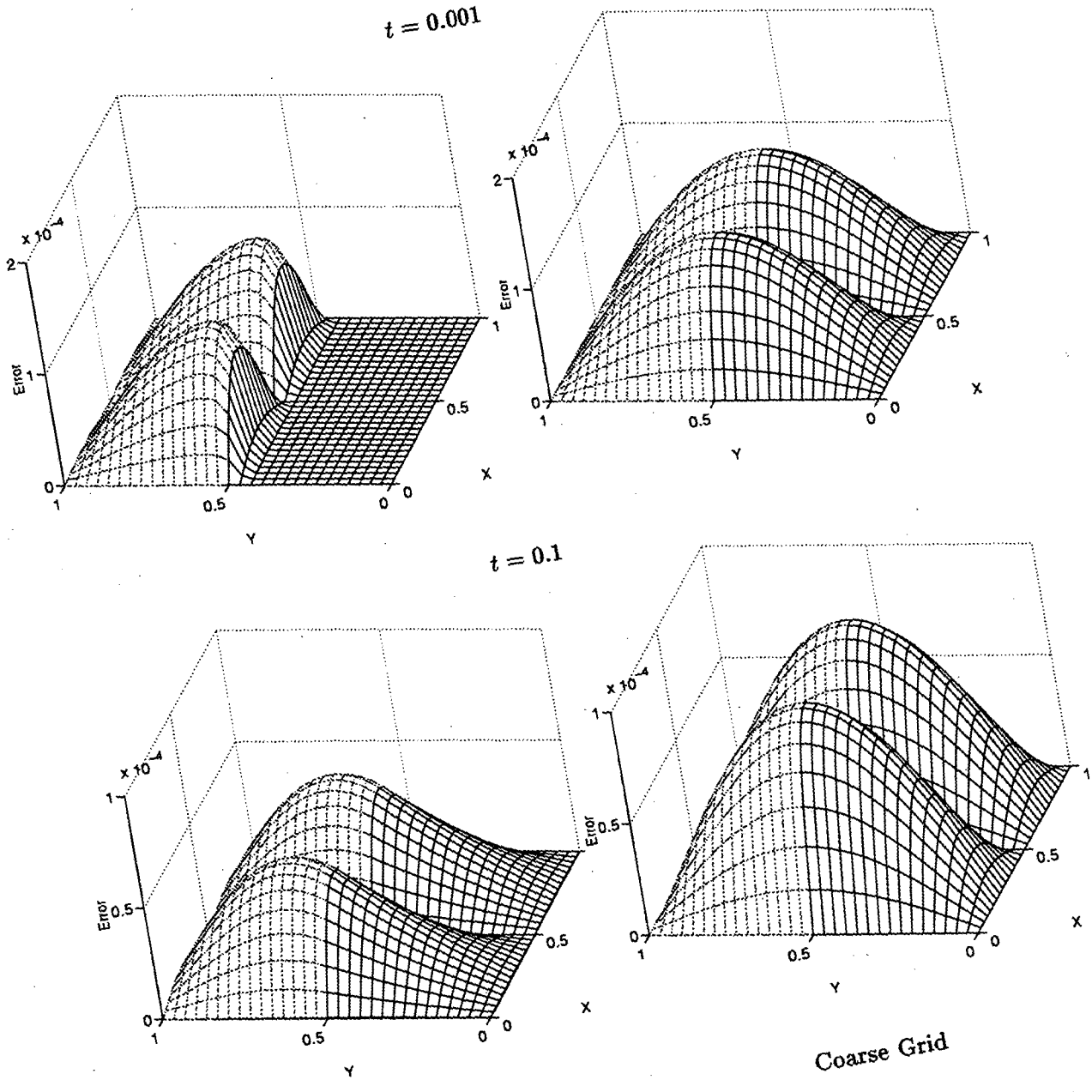


Figure 4.15: Comparison of Solutions for $k_1 = 2$, $k_2 = 1$

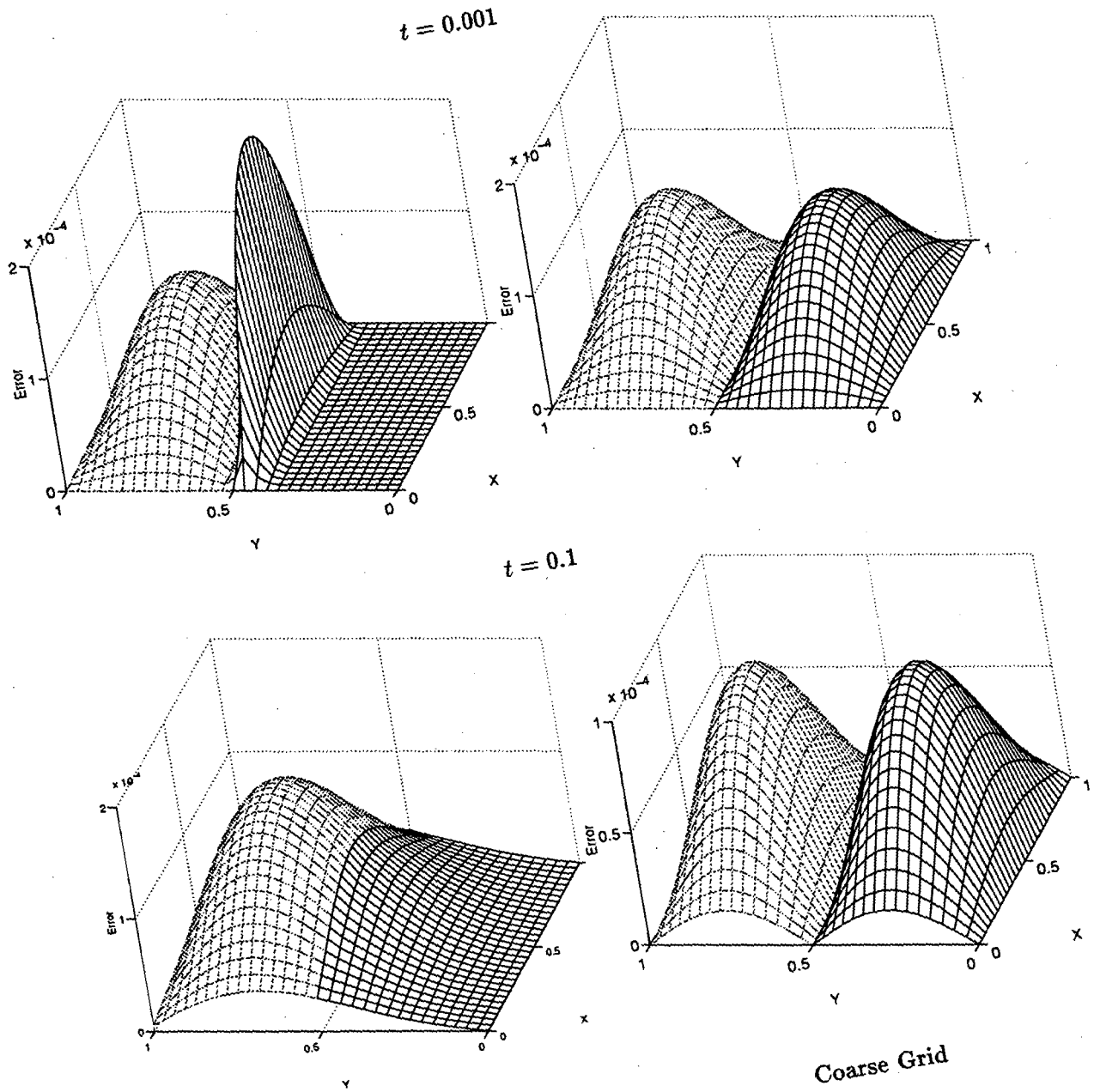
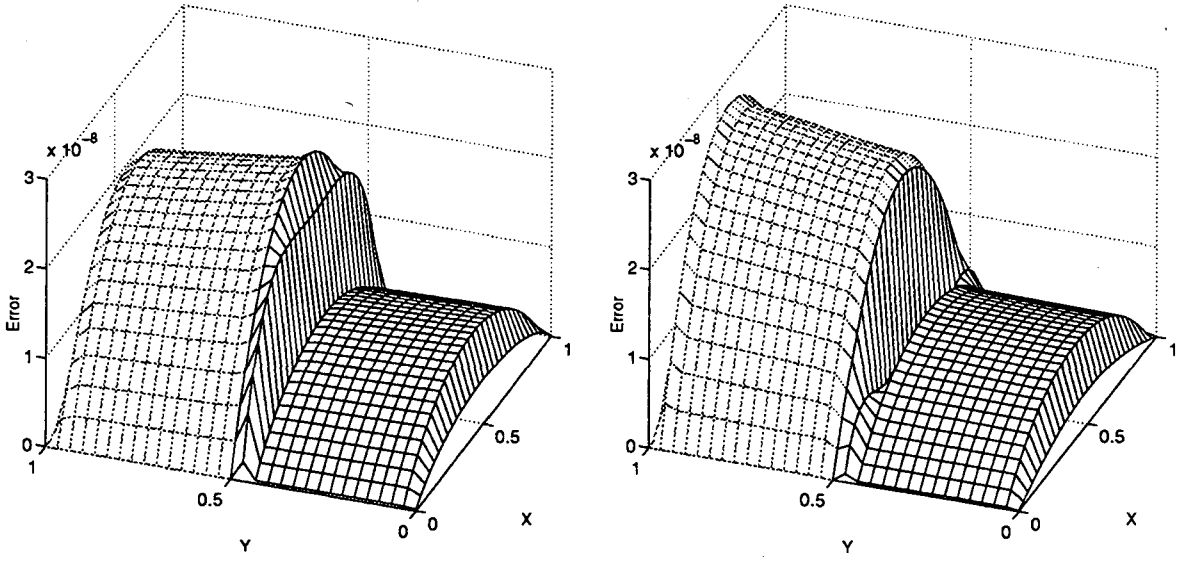


Figure 4.16: Comparison of Solutions for $k_1 = 1$, $k_2 = 2$

The equation for motion by curvature proves even less receptive to this scheme. Here the only test problem with an analytic expression for the solution is the collapse of circles under curvature as given in §1.4. The variable parameter is that of X_0, the point about which the circles are centered and to which they are collapsing. The relative position of this point from the center of the domain will be specified in polar form as (r, θ). This collection of test problems is a useful one as it broadly represents the effect of the adaptive mesh on all local orientations and curvatures. A scheme which performs well over the range of these parameters on the model problem will therefore be a promising candidate for development into a scheme for the general adaptive mesh problem. The current scheme, however, fails the test.

The difficulty can be qualified by considering the motion of the surfaces relative to the interface between the two grid resolutions and breaking it into two cases: motion normal to the interface, ($\mathbf{F} \perp \hat{\mathbf{x}}$), and motion tangential to the interface, ($\mathbf{F} \parallel \hat{\mathbf{x}}$). The

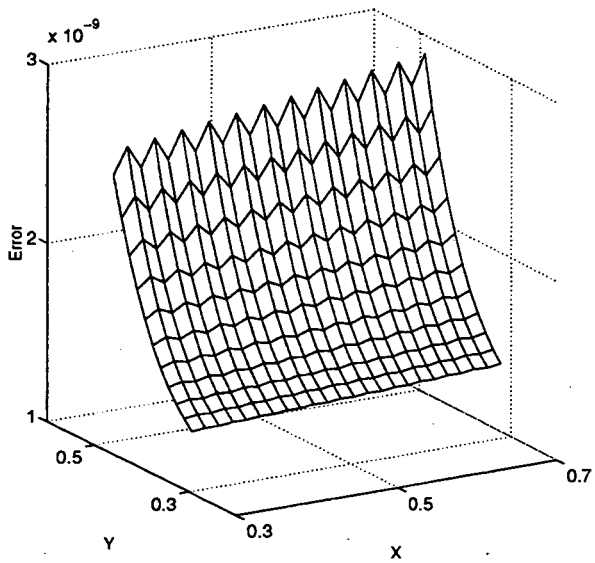


(a) Inward Flow (b) Outward Flow

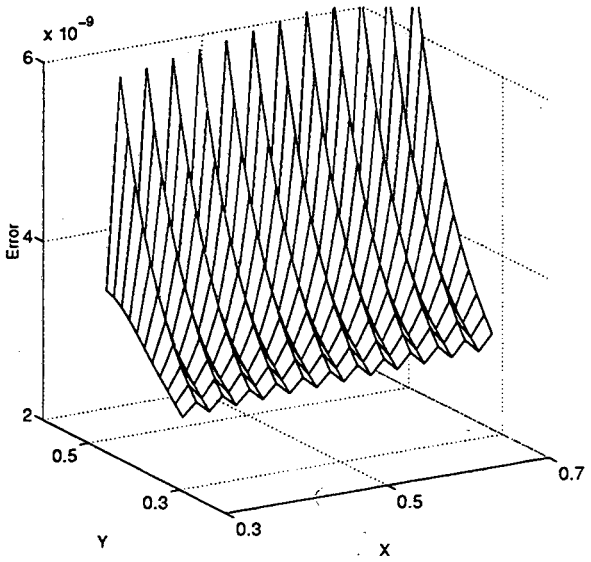
Figure 4.17: Flow of Surface Normal to Refinement Transition

latter case is represented by $\theta = 0$ and the former by $\theta = \pm\frac{\pi}{2}$, with r large. The errors at $t = 1.0$ for "normal flow" both into $(r = 10, \theta = -\frac{\pi}{2})$, and out of $(r = 10, \theta = \frac{\pi}{2})$, the fine mesh are shown in Fig–4.17. The scheme performs acceptably under both of these conditions.

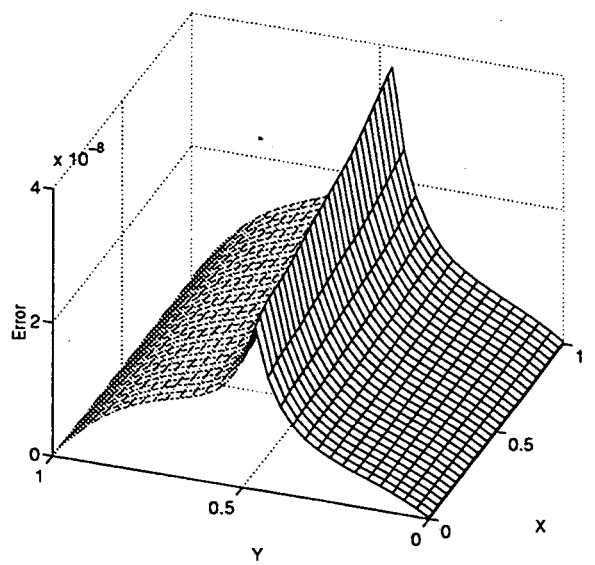
It is during the tangential flow of $(r = 10, \theta = 0)$ that the difficulties become apparent. By watching the error on the fine mesh, it is possible to see what has gone



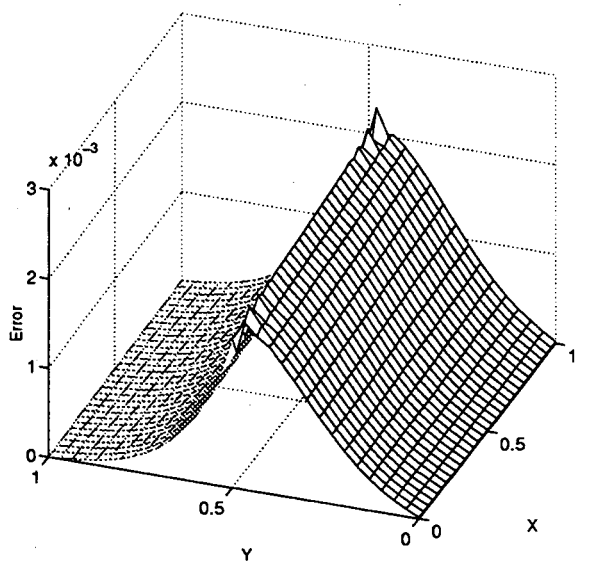
(a) Fine Grid at $t=0.01$



(b) Fine Grid at $t=0.02$



(c) At $t=0.05$



(d) At $t=0.1$

Figure 4.18: Breakdown of Solution under Tangential Flow

wrong. A close-up of the error in the fine cells is shown at $t = 0.01$ and $t = 0.02$ in Fig–4.18a-b. In these, the development and growth of oscillations can be seen (much like those for the interpolative scheme of Fig–4.8). Soon, the resultant error grows to the point where it becomes the dominant feature in Fig–4.18 at $t = 0.05$, and at $t = 0.1$ it has grown by five more orders of magnitude. The growth of this error term is reminiscent of the artificial source term seen in the heat equation, but in this case refinement only serves to expedite

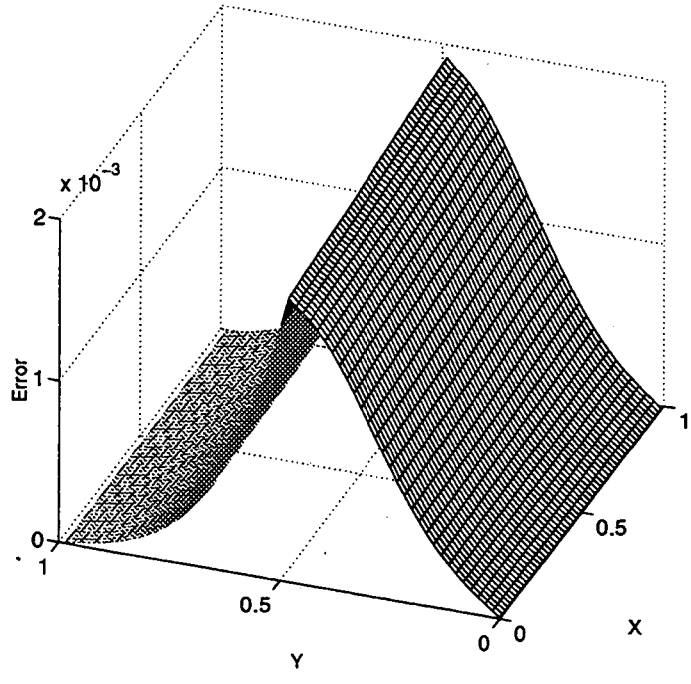


Figure 4.19: Effect of Refinement on Tangential Flow

the onset of difficulties. In Fig–4.19, the same problem is revisited at $t = 0.05$ but with the resolution of the grid doubled; here, the error has already reached the scale seen at $t = 0.1$ before. It is tempting to attribute the oscillatory behavior to the alternating "parity" of the stencil shown in Fig–4.14, but that is not the whole story, as will soon be shown.

Before moving on toward a resolution of these difficulties, it should first be established that they are truly significant. They have been found under the paradigm that for any test problem on any mesh, refining cells should never make the solution worse. In some of the examples cited as difficulties above (as in Fig–4.16), the L_2 norm of the error decreases over the whole domain even though it increases at the transition in refinement. The mitigating factor could be argued that the specially constructed refinement patterns

and initial conditions above are artificial and serve no apparent purpose. Typically, the mesh will be refined where the error is relatively large and increased resolution is needed. If that is done, then the resulting improvement within the fine mesh is likely to supersede any difficulties along its periphery. Such an effect is, no doubt, often the case, and that probably accounts for the success of similar methods [3] in reducing the L_2 norm of the error for specific problems, but such limited successes in no way guarantee an universal applicability.

For the evolution equation which originally motivated this discussion, i.e. for flow under curvature, the failure is quite relevant. The scheme fails when the flow is directed along (and not across) the boundary between mesh resolutions. Since the "flow" is normal to the level surfaces, difficulties can be expected whenever a transition in resolution is made along a level surface. Such a transition is often desired, as it is in the study of the topological change underway in Fig–3.3. Requiring, say, that the zero level set be contained in the finest level would greatly reduce the usefulness of the method (and take it to the realm where a narrow band scheme [2] would be equally applicable and often more appropriate). The difficulties are, therefore, of the upmost significance and must be resolved in order to have a useful method for solving the level set equation on an adaptive mesh.

4.5 Resolution

At this point, there are several clear complaints about the spline based method proposed above. First, it certainly lacks sufficient accuracy to be of any general use, and although it gives an improvement over an interpolative scheme in some problems, it still seems to suffer from the same type of artificial source term in others. Second, it requires the provision of an "appropriate" stencil without providing a clear definition or a clear means for establishing such a stencil. Third, and as previously mentioned, the stencil may not be "centered" about the evaluation point.

These issues are not unrelated; the uncentered stencil of the last, for example, could generate the type of artificial source cited in the first by enacting an implicit interpolation from the center of the stencil to the desired evaluation point. It is useful, however, to consider them separately in order to systematically attack the overall difficulty.

The second issue is dodged by the model problem in which the two or three required stencil types can be handled explicitly. Later, for a fully adaptive mesh (especially in three dimensions) the number of stencils will proliferate and an algorithm for their selection will be essential. The construction of such an algorithm to choose five (or, in three dimensions, eight) points is worrisome, mainly because of situations in which there is no symmetric choice can be made. The choice of the final stencil point in Fig–4.20, for example, is unclear.

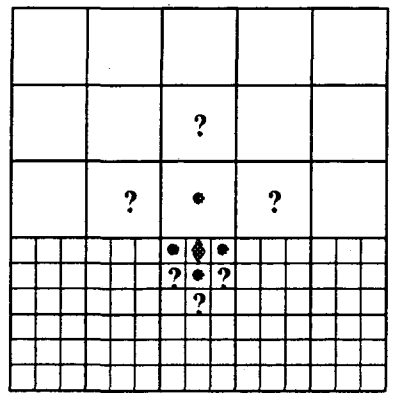


Figure 4.20: An Unclear Choice of Stencil

The critical realization here is that it is not difficult to generate candidates for stencil points — the difficulty is in choosing the requisite number from the pool of candidates,

especially when that cannot be accomplished in a symmetric fashion. To avoid adding an effectively random, asymmetric choice (or trying to determine, locally, which asymmetric choice would be "best"), the entire pool of ($n \geq 5$) candidates could be included in the stencil, with Eq–4.3 becoming the overdetermined system

$$\underbrace{\begin{pmatrix} x_1 & y_1 & {x_1}^2 & {y_1}^2 & x_1y_1 \\ x_2 & y_2 & {x_2}^2 & {y_2}^2 & x_2y_2 \\ x_3 & y_3 & {x_3}^2 & {y_3}^2 & x_3y_3 \\ x_4 & y_4 & {x_4}^2 & {y_4}^2 & x_4y_4 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & {x_n}^2 & {y_n}^2 & x_ny_n \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} b_1 \\ b_2 \\ c_{11} \\ c_{22} \\ c_{12} \end{pmatrix}}_{\mathbf{p}} = \underbrace{\begin{pmatrix} \Delta\varphi_1 \\ \Delta\varphi_2 \\ \Delta\varphi_3 \\ \Delta\varphi_4 \\ \vdots \\ \Delta\varphi_n \end{pmatrix}}_{\varphi} \tag{4.4}$$

for the spline, $\mathbf{p}$, which can now be fit to the expanded stencil as the least squares solution:

$$\|\mathbf{Ap} - \varphi\|_2 = \min_q \|\mathbf{Aq} - \varphi\|_2 .$$

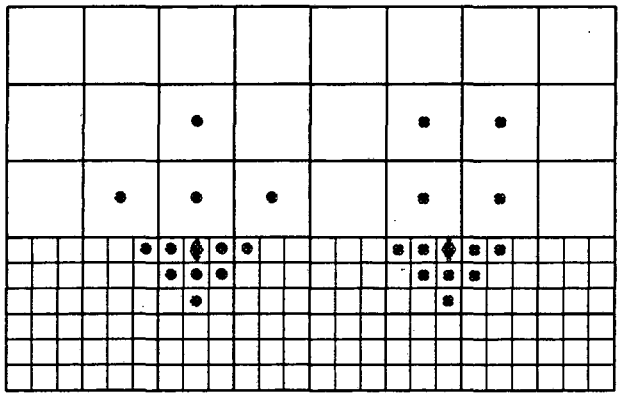


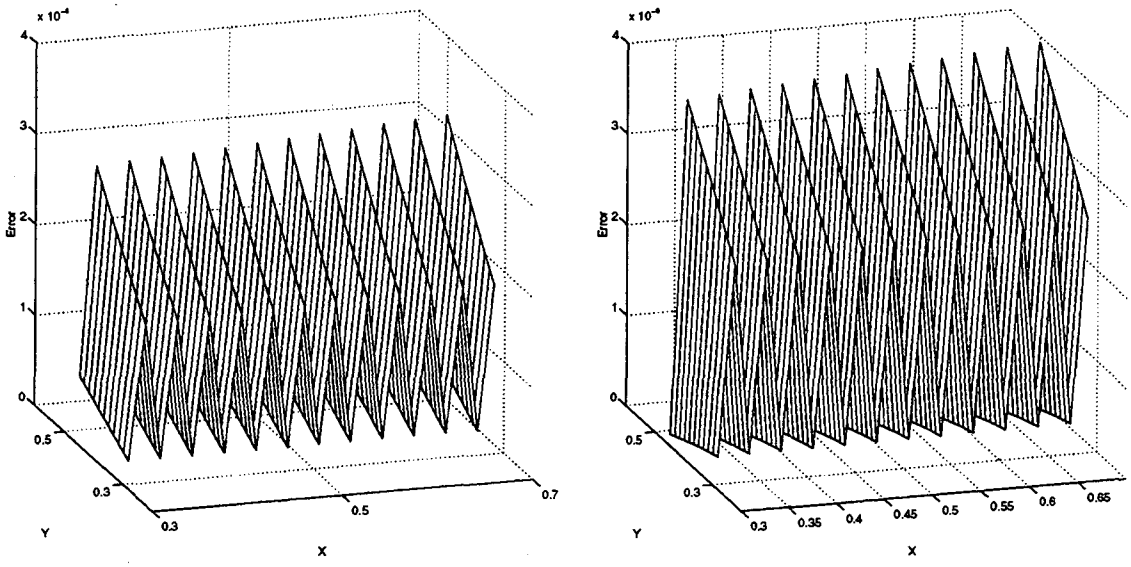
Figure 4.21: Stencils for the Improved Method

In addition to allowing a more natural choice of stencil points, the overdetermination of the system also offers an additional degree of freedom which can be exploited to great advantage with respect to centering the spline on the evaluation point. Since the spline is no longer being forced to fit the stencil data exactly, it can be made to fit some data points more closely than others by weighting the system, $(\omega\mathbf{A})\mathbf{p} = \omega\varphi$, where ω is a $n \times n$ diagonal matrix, and then finding the weighted least squares solution:

$$\|\omega(\mathbf{Ap} - \varphi)\|_2 = \min_q \|\omega(\mathbf{Aq} - \varphi)\|_2 .$$

In particular, the fit can be weighted according to the distance from the evaluation point, $\omega_{ii} = ({x_i}^2 + {y_i}^2)^{-1/2}$, so that the spline agrees closely with the nearby points, but only losely with the more distant points.

Making this generalization greatly simplifies the requirements on selecting a stencil. There are only two obvious restrictions which must be met: that the points be generally nearby (if all the points are far away the weighting cannot help), and that they be reasonably distributed in location (if they are all to one side the evaluation will remain uncentered). It is natural to hope that such a construction will relieve the oscillation seen in Fig–4.18 and lead to a successful method. It does help, and for many problems that is probably sufficient, but for curvature flow it is not. For the above tangential flow problem, oscillations still develop in the fine grid as shown in Fig–4.22 at the times of $t=0.05$ and $t=0.1$. The oscillations do not grow as quickly as they did in the original method, but they still eventually cause the deterioration of the solution.



(a) Fine Grid At $t=0.05$ (b) Fine Grid At $t=0.1$

Figure 4.22: Persistence of Oscillations under Tangential Flow

Presuming, then, that the second and third complaints cited at the beginning of this section have been answered, it is only the first that remains — and that is the question of accuracy. The assumption is then that the spline construction is accomplishing something akin to the uniform grid central difference operator, but that it must not attain the same

accuracy. The inaccuracy is probably related to the (directionally unbalanced) distribution of the stencil points. It may be that there is an as yet undiscovered algorithm for generating stencils that overcomes this difficulty, but there is another, general remedy at hand. This is to include the next degree terms in the spline for the purpose of improving the least squares solution (even though they themselves need not be computed):

$$\underbrace{\begin{pmatrix} x_1 & y_1 & x_1^2 & y_1^2 & x_1y_1 & x_1^3 & y_1^3 & x_1^2y_1 & x_1y_1^2 \\ x_2 & y_2 & x_2^2 & y_2^2 & x_2y_2 & x_2^3 & y_2^3 & x_2^2y_2 & x_2y_2^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & x_n^2 & y_n^2 & x_ny_n & x_n^3 & y_n^3 & x_n^2y_n & x_ny_n^2 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ d_{122} \end{pmatrix}}_{\mathbf{p}} = \underbrace{\begin{pmatrix} \Delta\varphi_1 \\ \Delta\varphi_2 \\ \vdots \\ \Delta\varphi_n \end{pmatrix}}_{\varphi} \tag{4.5}$$

Identifying the coefficients of the spline with its derivatives, the inclusion of these third order terms effectively allows for the spatial interpolation of the second order terms, the c_{ij}'s, within the stencil. This "interpolation" dispels the concern over the directional imbalance of the stencil, and in fact produces the type of results that have been hoped for all along.

The following results are for stencils (which now must contain at least nine points) that have been chosen as depicted in Fig–4.21. This figure shows a refinement ratio of three, but the choice is similar for all refinement ratios. The actual algorithm used to generate these stencils will be described in §4.6 in the context of a fully adaptive three-dimensional mesh.

Revisiting the problem case for the heat equation with $k_1 = 1$ and $k_2 = 2$ the difficulties along the periphery is seen to be resolved by the improved scheme, as shown at $t = 0.001$ in Fig–4.24. Furthermore, it performs nicely under curvature flow in various directions, 124568 ($r = 1, \theta = k\frac{\pi}{8}$) for $k = -4, -3, -1, 0, 2, 3$; the errors for which are depicted in Fig–4.23, each at $t=1.0$.

At this point, nothing has been said about how the least squares problem is to be solved numerically or how much it costs or how it can be done efficiently. Those issues have been intentionally left out in order to focus attention on the accuracy and the basic usefulness of the method. Chapter 5 will concern itself with the various fine points of actual implementation. But first it is time to leave the model problem behind and try out the method on the real three-dimensional adaptive mesh.

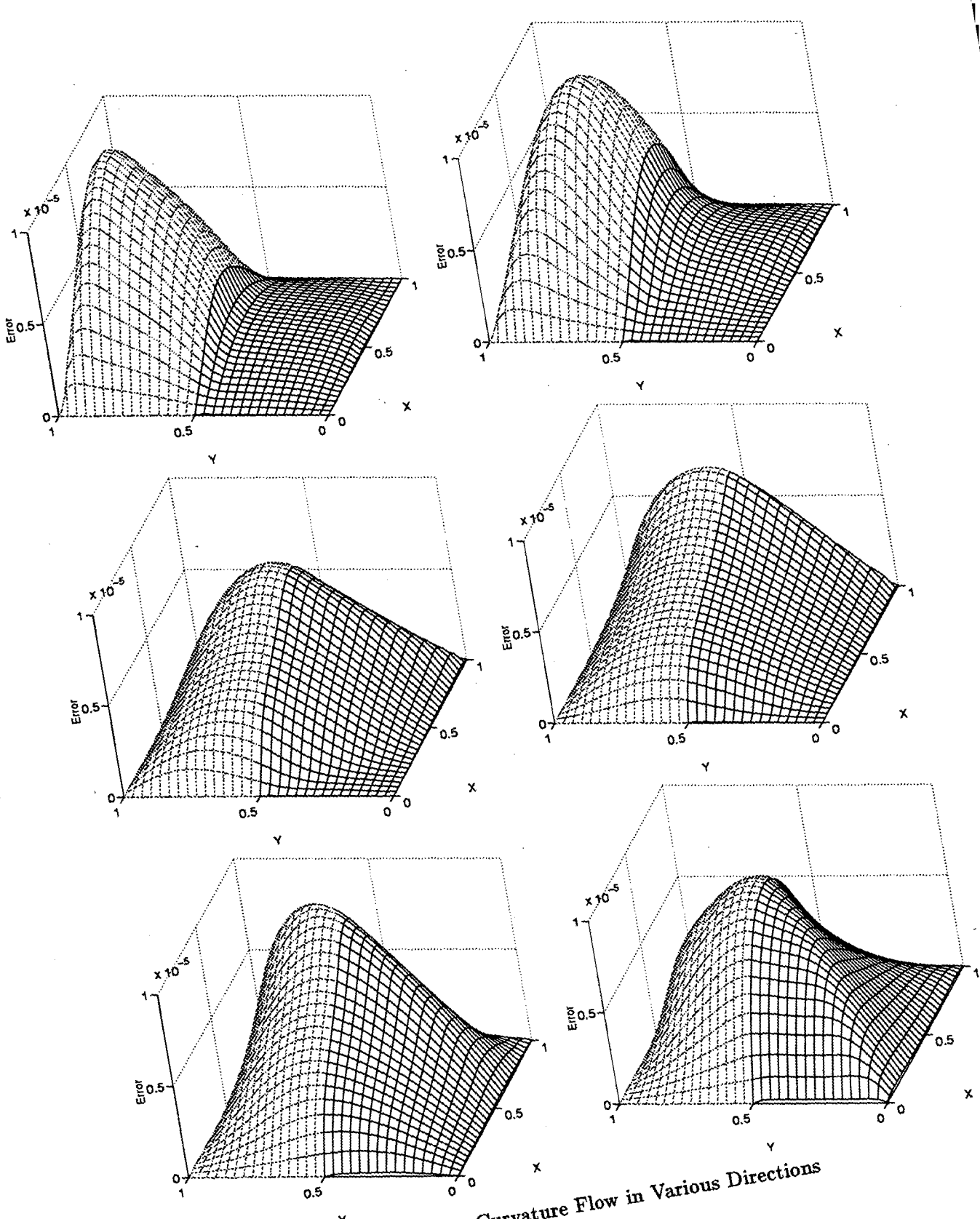


Figure 4.23: Curvature Flow in Various Directions

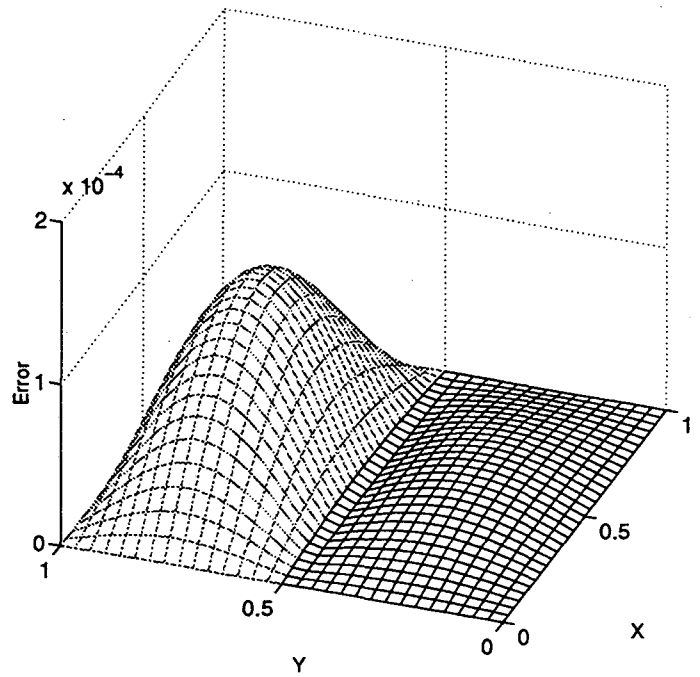


Figure 4.24: Resolution of Difficulty for the Heat Equation

4.6 Return to Three Dimensions

The above machinery (Eq–4.5) carries over directly to three dimensions, with the addition of the appropriate terms for the generic third order polynomial in three dimensions:

$$\underbrace{\begin{bmatrix} x_1 & y_1 & z_1 & {x_1}^2 & \dots & x_1y_1z_1 \\ x_2 & y_2 & z_2 & {x_2}^2 & \dots & x_2y_2z_2 \\ x_3 & y_3 & z_3 & {x_3}^2 & \dots & x_3y_3z_3 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_n & y_n & z_n & {x_n}^2 & \dots & x_ny_nz_n \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ d_{123} \end{bmatrix}}_{\mathbf{p}} = \underbrace{\begin{bmatrix} \Delta\varphi_1 \\ \Delta\varphi_2 \\ \Delta\varphi_3 \\ \vdots \\ \Delta\varphi_n \end{bmatrix}}_{\varphi} \tag{4.6}$$

Aside from its size, this is no different than its two-dimensional counterpart. The only issue, then, for implementing the method in three dimensions, or for that matter, on a fully adaptive mesh in any dimension, is the formulation of an algorithm to generate stencils. The algorithm will be of maximal use if it places no special requirements on the structure of the mesh in terms of refinement ratios or the nesting of levels. For exceedingly large refinement ratios (or, correspondingly, very poorly nested levels), the above spline-based

method will generally fail because of inhomogeneity in stencil spacing, regardless of the algorithm used to generate the stencil. The test of a "good" algorithm is then the degree of disparity in resolution which it can accommodate before breaking down. What will now be described is a particularly robust algorithm for selecting stencils.

The basic tools this algorithm uses are generalized versions of the shift operators defined in §3.1. These will be denoted $\mathbf{G_d}$, where $\mathbf{d}$ is one of the coordinate directions, $\mathbf{d} \in \mathbf{D} = \{\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}, -\hat{\mathbf{x}}, -\hat{\mathbf{y}}, -\hat{\mathbf{z}}\}$. They are generalizations of the shift operators in that $\mathbf{G_d}(C) = \mathbf{S_d}(C)$ wherever $\mathbf{S_d}(C) \neq \emptyset$, but they always map cells to cells, so $\mathbf{G_d} : \mathcal{M} \rightarrow \mathcal{M}$. For a given cell and direction, $\mathbf{G_d}(C)$ is defined as being the finest cell which abuts C on its $\mathbf{d}^{\text{th}}$-face and satisfies $\mathbf{Lev}(\mathbf{G_n}(C)) \leq \mathbf{Lev}(C)$, i.e. that is no finer that C. This definition can be written explicitly as

$$\mathbf{G_d}(C) \;=\; \mathbf{S_d}\left(\mathbf{Parent}^k(C)\right) \quad \text{where} \quad k = \min\left\{k' \geq 0 :\; \mathbf{S_d}\left(\mathbf{Parent}^{k'}(C)\right) \neq \emptyset\right\} . \tag{4.7}$$

That such a cell must exist and be unique follow immediately from the definition since

$$\mathbf{S_d}\left(\mathbf{Parent}^i(C)\right) \neq \emptyset \quad \text{where} \quad C \in L_i .$$

These operators can be envisioned as a means of flowing from cell to cell and from fine cells to coarser cells where necessary (but never from coarse to fine); a pictorial representation is given in Fig–4.25. Because of the bias from coarse to fine, it is possible for

$$\mathbf{G_{d_2}} \circ \mathbf{G_{d_1}}(C) \neq \mathbf{G_{d_1}} \circ \mathbf{G_{d_2}}(C) , \quad \text{and for} \quad \mathbf{G_{-d}} \circ \mathbf{G_d}(C) \neq C .$$

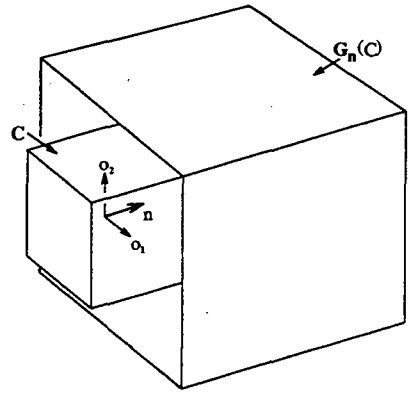


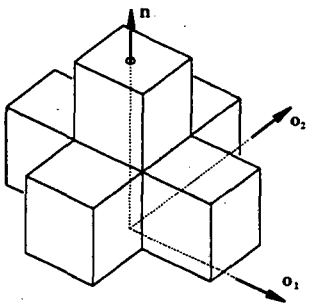
Figure 4.25: The Generalized Shift Operator

The algorithm to construct the stencil uses these generalized shift operators to collect the stencil cells. For each $\mathbf{n} \in \mathbf{D}$, it chooses coordinate directions $\mathbf{o_1(n)}, \mathbf{o_2(n)} \in \mathbf{D}$

such that $\{\mathbf{n}, \mathbf{o_1}, \mathbf{o_2}\}$ form a basis, as also shown in Fig–4.25. The projections of the vector $\mathbf{X}(\mathbf{G_n}(C)) - \mathbf{X}(C)$ onto $\mathbf{o_1}$ and $\mathbf{o_2}$ are denoted as x_1 and x_2, and the stencil is defined as consisting of all cells listed in the appropriate case below for each choice of $\mathbf{n}$ — the sketches at the right depict the cells that will be selected in the event that they come from a single level.

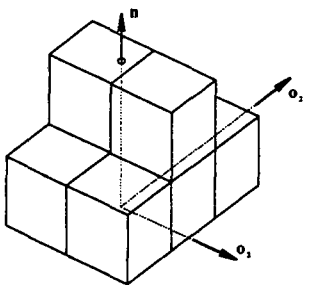
- If $x_1 = x_2 = 0$:

$\mathbf{G_n}(C) \equiv N$	$\mathbf{G_n}(N)$
$\mathbf{G_{o_1}}(N)$	$\mathbf{G_{-o_1}}(N)$
$\mathbf{G_{o_2}}(N)$	$\mathbf{G_{-o_2}}(N)$



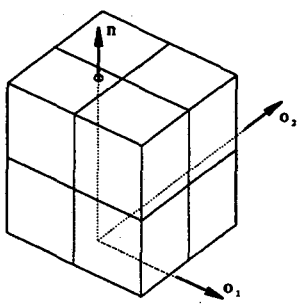
- If $x_1 > 0$, $x_2 = 0$: (Same list if $x_1 = 0$, $x_2 > 0$, with $\mathbf{o_1}$ and $\mathbf{o_2}$ interchanged.)

$\mathbf{G_n}(C) \equiv N$	$\mathbf{G_n}(N)$
$\mathbf{G_{o_2}}(N)$	$\mathbf{G_{-o_2}}(N)$
$\mathbf{G_{o_1}}(N) \equiv N_2$	$\mathbf{G_n}(N_2)$
$\mathbf{G_{o_2}}(N_2)$	$\mathbf{G_{-o_2}}(N_2)$



- If $x_1 > 0$, $x_2 > 0$:

$\mathbf{G_n}(C) \equiv N$	$\mathbf{G_n}(N)$
$\mathbf{G_{o_1}}(N) \equiv N_2$	$\mathbf{G_n}(N_2)$
$\mathbf{G_{o_2}}(N) \equiv N_3$	$\mathbf{G_n}(N_3)$
$\mathbf{G_{o_2}}(N_2) \equiv N_4$	$\mathbf{G_n}(N_4)$



- If $x_1 < 0$ (respectively $x_2 < 0$) the corresponding case above for $x_1 > 0$ applies with $\mathbf{o_1}$ replaced by $-\mathbf{o_1}$.

After these cells have been collected, any cell in the stencil that also has a descendent in the stencil is removed. These situations can occur because of eventualities such as $\mathbf{G_{d_2}} \circ \mathbf{G_{d_1}}(C) = \mathbf{Child} \circ \mathbf{G_{d_1}} \circ \mathbf{G_{d_2}}(C)$.

On a fully adaptive mesh, this algorithm produces a wide variety of stencils, especially if the mesh is less than 2-nested. The prototypical stencil for a fine cell on the face of an expanse of fine cells is shown in Fig–4.26. The cubes mark the cell centers, with the larger cubes on the bottom corresponding to the coarse cells in the stencil. Other typical stencils for cells near a corner of a block of fine cells are shown in Fig–4.27 and Fig–4.28. All of these stencils are taken from a mesh with a refinement ratio of three.

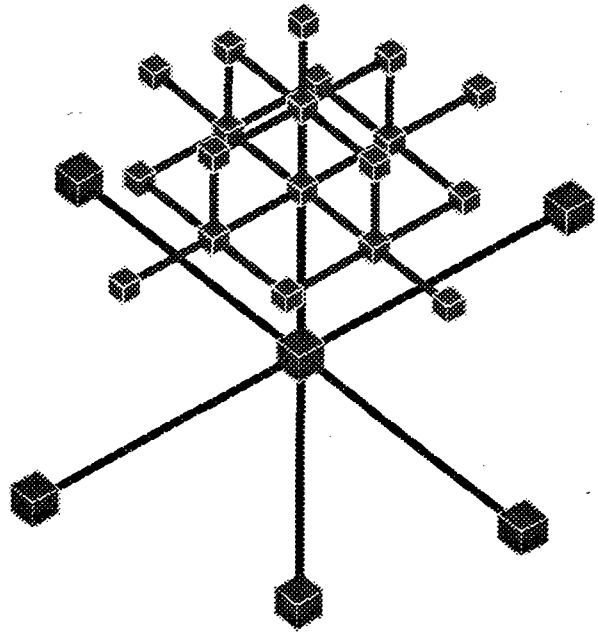


Figure 4.26: Face Type Stencil

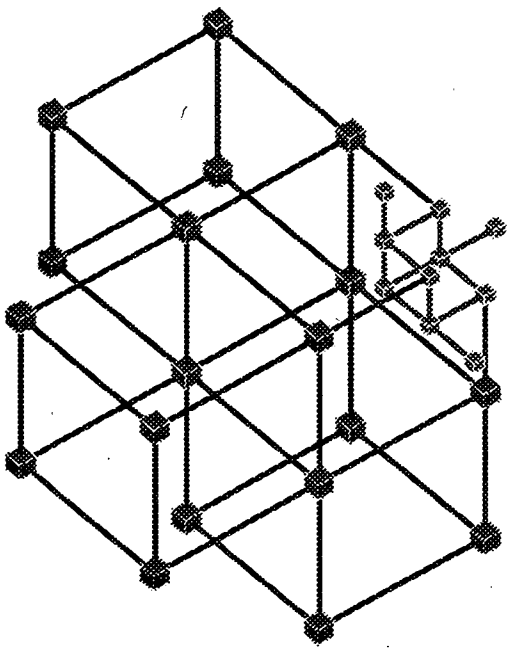


Figure 4.27: Corner Type Stencil

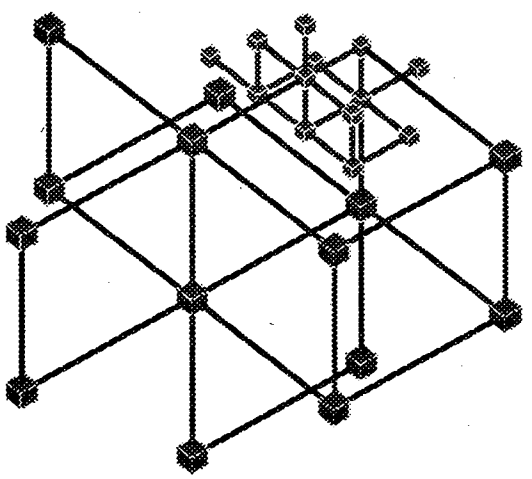


Figure 4.28: Stencil for Cell Adjacent to Corner

4.7 Results

Under this algorithm the method can now be tried on the two test problems introduced in §4.1 where the mesh is refined in either a band or a patch. For the first problem, the scheme performs admirably, giving a well defined reduction in error along the zero level surface. For the collapse of a unit sphere under curvature, with a single band of refinement, the (log) errors from a central slice are plotted in the sequence Fig–4.29 to Fig–4.31. These are taken at the early time of $t = 0.1$ the intermediate time of $t = 0.3$ and at the time of collapse (for the zero level surface) of $t = 0.5$. Fig–4.32 gives a cutaway version of the error at $t = 0.3$ to better depict the error within the band. Throughout the computation, refinement provides a clear benefit in the band, with errors well below that of the coarse grid. In fact, the error is virtually identical to the corresponding uniform fine grid computation. This is illustrated in Fig–4.33 where the increase in error over the full method is plotted at $t = 0.3$.

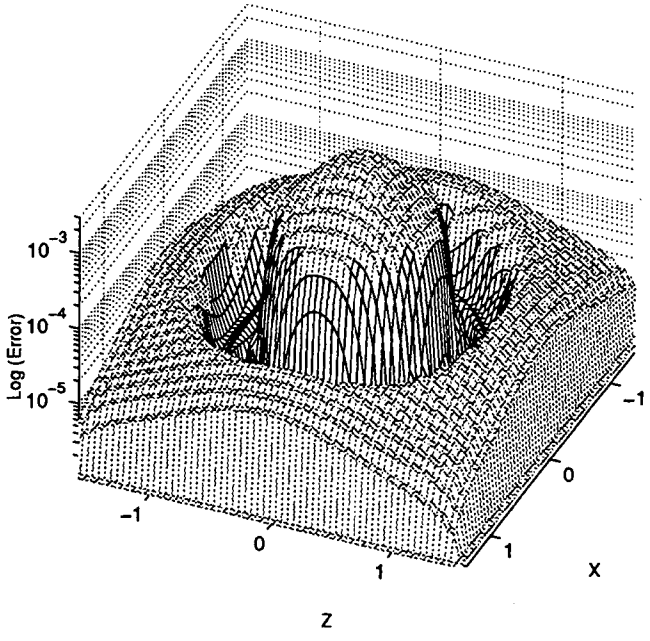


Figure 4.29: Error under Banded Refinement at $t = 0.1$

The test of greater concern is the one for which the refinement is restricted to a patch on the zero level surface, and that test was originally introduced in anticipation of

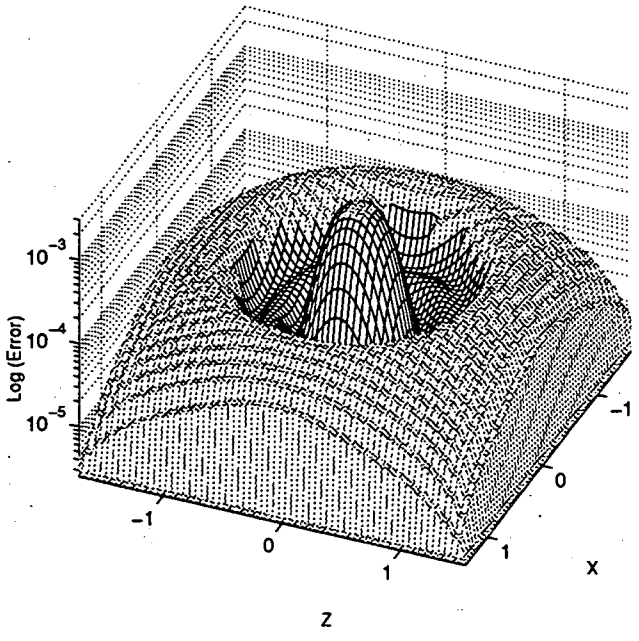


Figure 4.30: Error under Banded Refinement at $t=0.3$

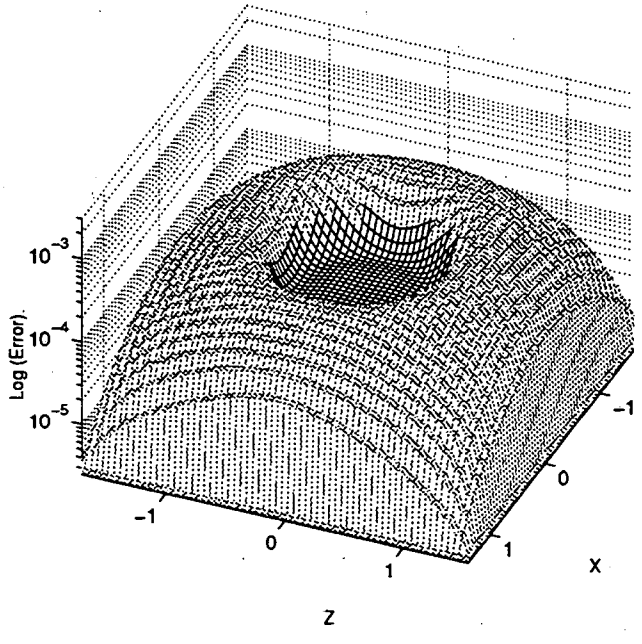


Figure 4.31: Error under Banded Refinement at $t=0.5$

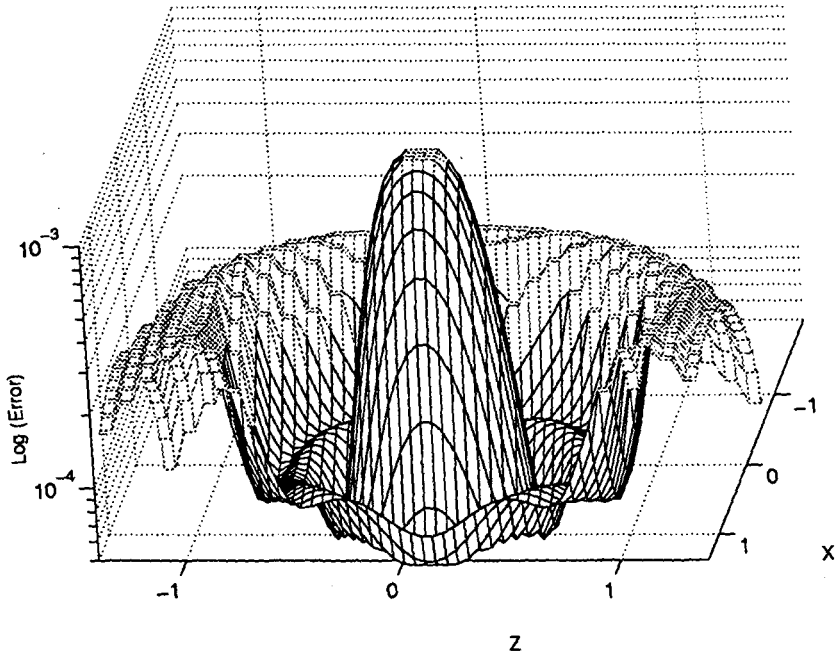


Figure 4.32: Cut-Away of Error under Banded Refinement

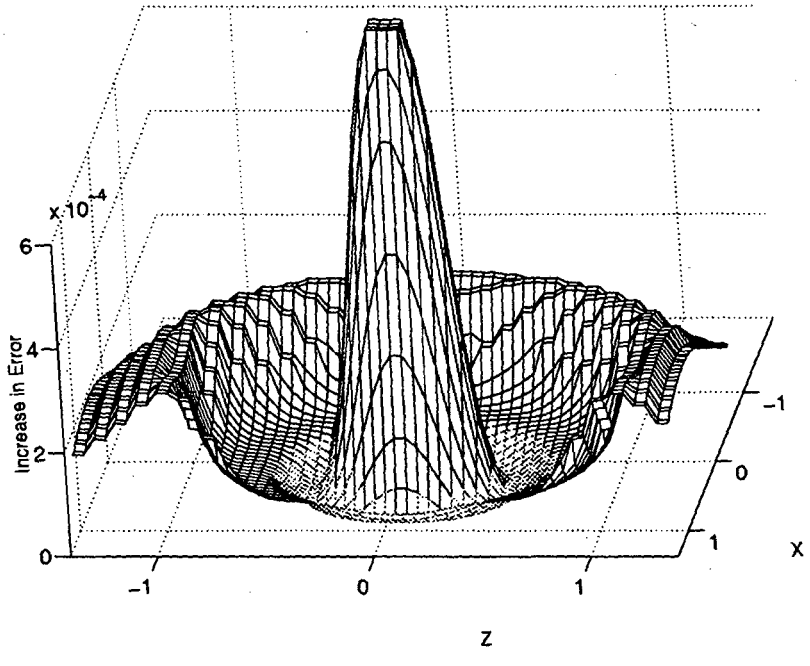


Figure 4.33: Comparison of Banded Refinement to Full Method

its usefulness in testing the parabolic scheme under refinement across the surface. For the adaptive method presented above, the behavior is again ideal. Figures Fig–4.34 to Fig–4.37 are the companion plots of Fig–4.29 to Fig–4.33 for this test. Here, the method can have no hope of maintaining a small error within the patch, since the diffusive nature of the evolution equation along the level surfaces will inevitable result in the encroachment of the error from the coarse cells into the fine. This increase of the error within the patch is illustrated by Fig–4.36. The important condition being tested is that the solution is not made worse, through the addition of the fine patch, than the uniform coarse grid solution. This method solidly meets that condition, as can be seen in the series of figures, with the error smoothly transitioning to the coarse grid level.

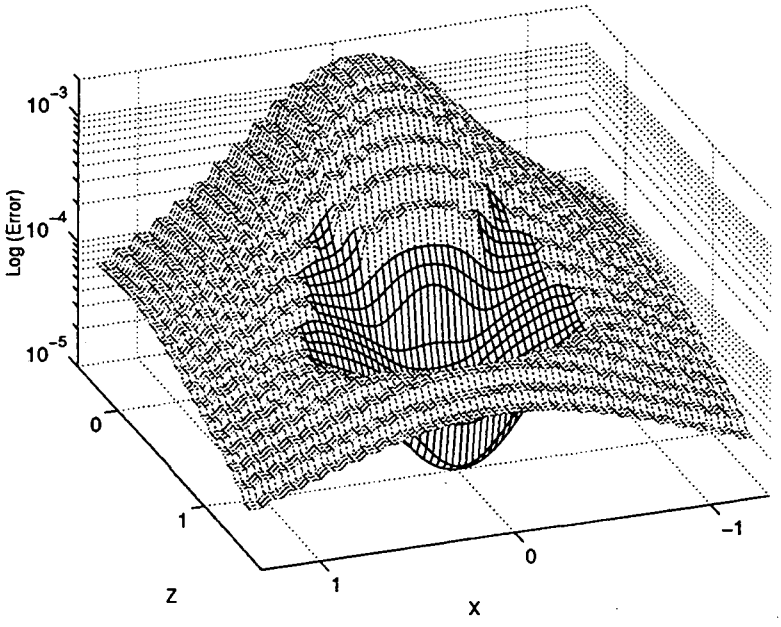


Figure 4.34: Error under Patched Refinement at $t=0.1$

A more stringent test can be provided by allowing the patch to be originally generated as above but then keeping it fixed throughout a lengthy computation. As the level surfaces continuously propagate through the patch, the error should be carried from the coarse cells into the fine patch and eventually become uniform. Should any oscillations develop as a result of the method used on the periphery of the patch, they will be reinforced by the fixed nature of the mesh and quickly become obvious.

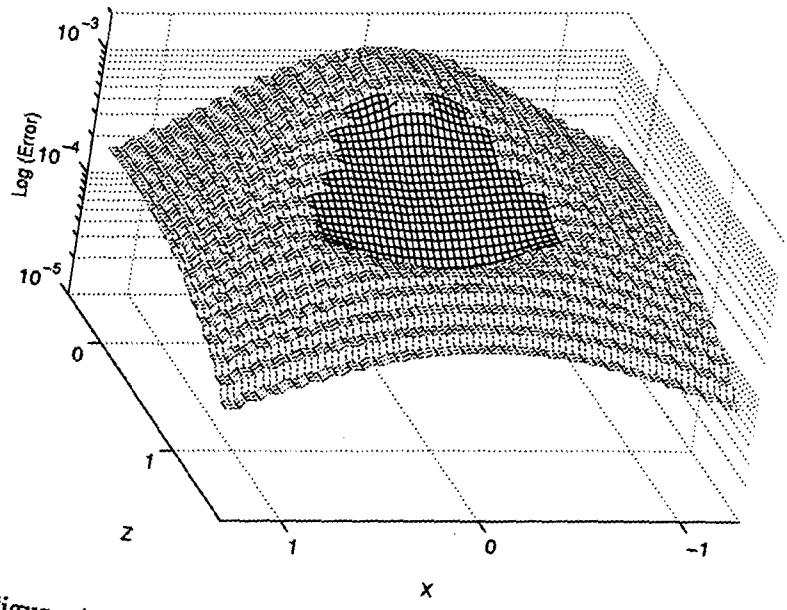


Figure 4.35: Error under Patched Refinement at $t=0.3$

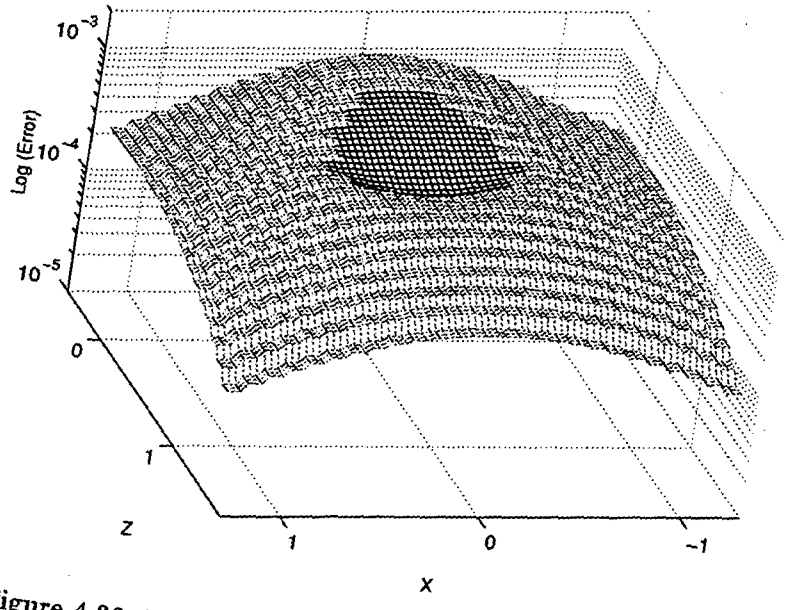


Figure 4.36: Error under Patched Refinement at $t=0.5$

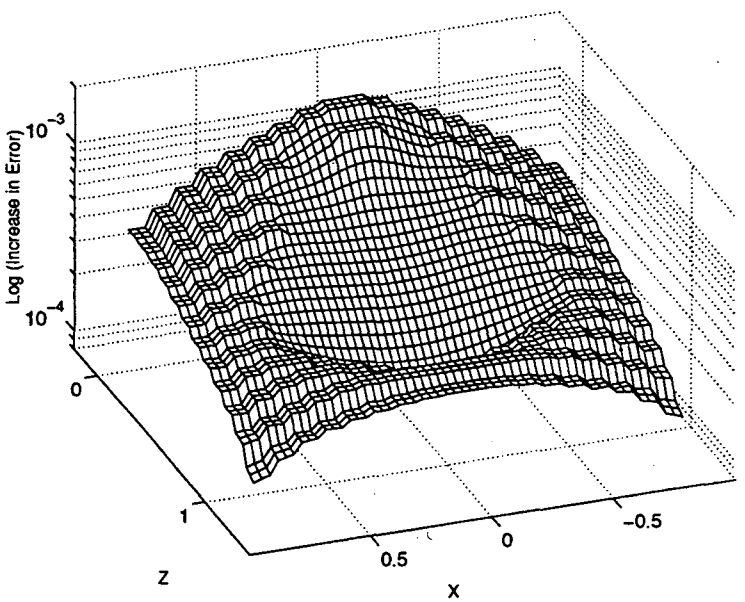


Figure 4.37: Comparison of Patched Refinement to Full Method

For a refinement ratio of three, Fig–4.38 shows the error in and around the patch at $t = 0.003$, 0.01, 0.03, 0.1, 0.3, and 1.0. The desired smooth transition is again seen in this case.

By conducting the same test with higher refinement ratios, it is possible to measure how well a given scheme can handle inhomogeneity in the mesh (including poor nesting). For the stencil generation algorithm of the previous section and the weighting of the least squares problem by inverse distance, the method is proven to be particularly robust. It works very well with refinement ratios as large as five, as is illustrated in the error plot, in Fig–4.39, from the steady-state time of $t=1.0$. The transition point at which the behavior begins to deteriorate is around a refinement ratio of six. Two error plots for this refinement ratio are shown in Fig–4.40. At $t=0.03$ oscillations begin to appear along the edges of the patch. By $t=0.1$ the error has grown within the patch and begun to corrupt the surrounding coarse solution.

That the method can perform well at this level of discontinuity is quite remarkable. The other stencil selection and weighting algorithms that have been tested all perform considerably more poorly, although the reason for the discrepancy is not currently well un-

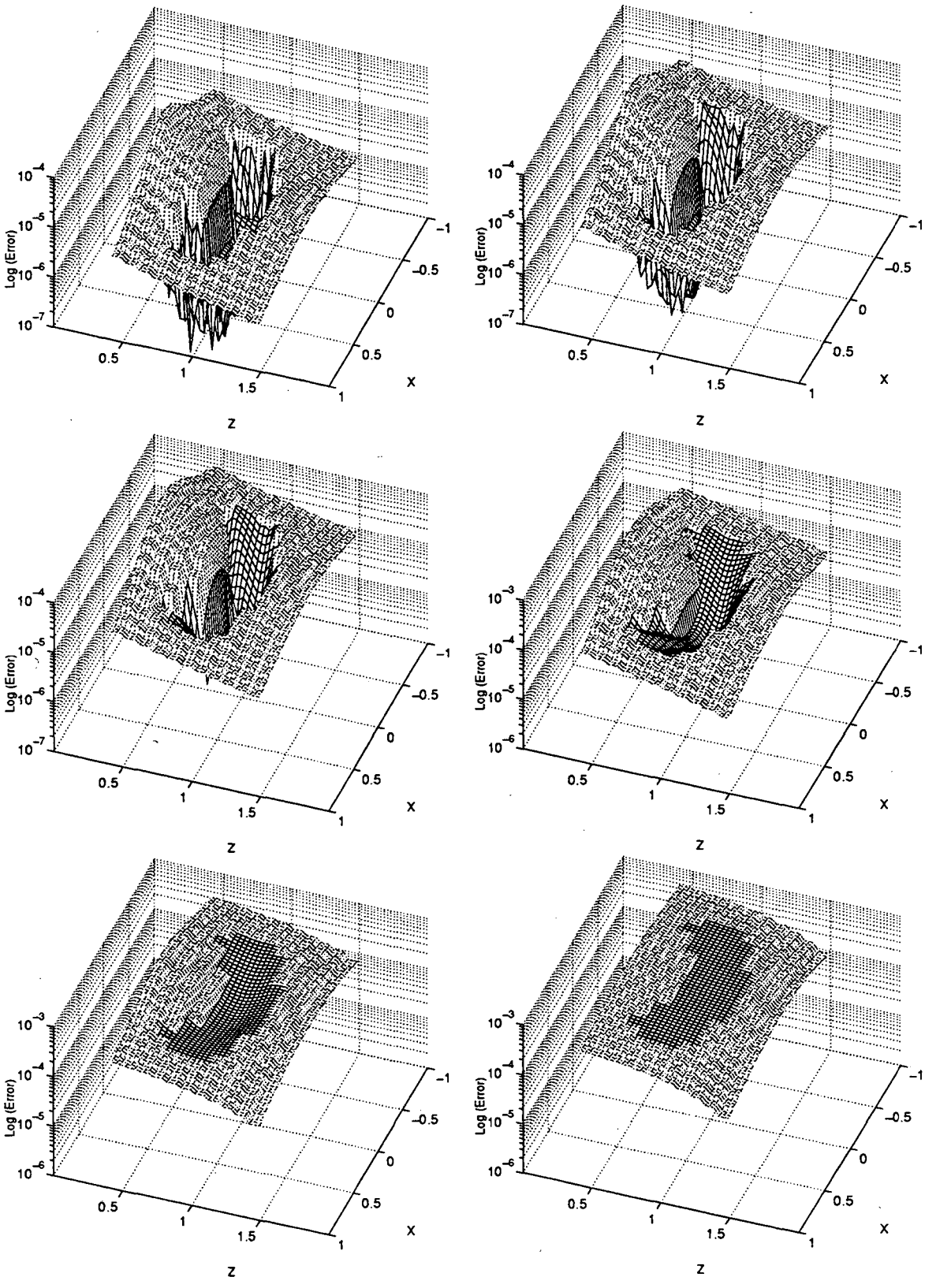


Figure 4.38: Accumulation of Error in a Fixed Patch

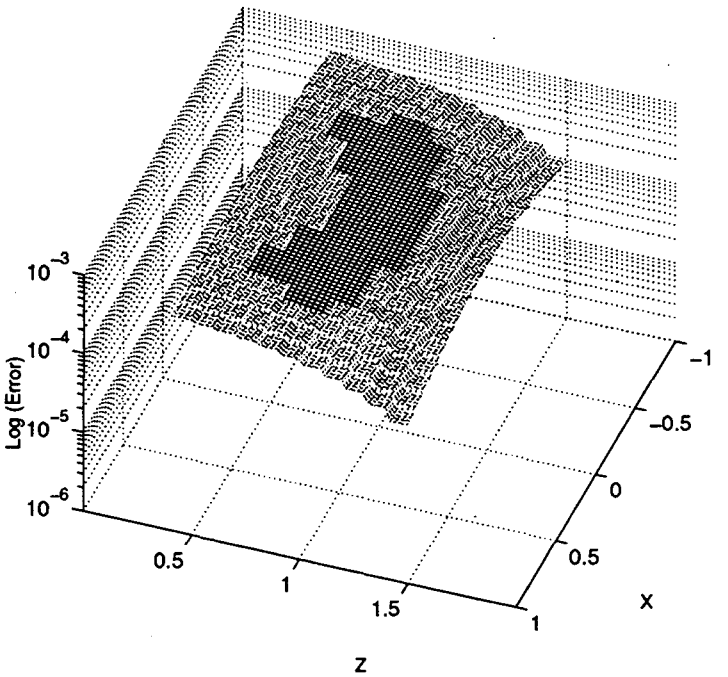


Figure 4.39: Long-Term Stability in a Fixed Patch with $rr = 5$

derstood. An argument for the optimal weighting method could possibly be derived directly from the least squares problem. Providing a similar argument for the stencil generation algorithm is probably a much more involved task. The existing implementation can easily accommodate any stencil generation algorithm that is based on the generalized shift operator.

4.8 Adaptive Time Stepping

The final concern, which has been intentionally disregarded up to this point, is the behavior of the method under the inclusion of adaptive time stepping on the levels. This inclusion does not generally cause a problem, with errors typically being decreased through its use. In, however, the combined presence of a parabolic (curvature) term and large jumps in refinement (because rr is large or because the levels are poorly nested), a difficulty is encountered. If the jump in refinement is given by rr, there will be a corresponding jump in time step on the order of rr^2. For sufficiently large jumps in time step, the spline based method will again begin to exhibit oscillatory behavior along the periphery, and these

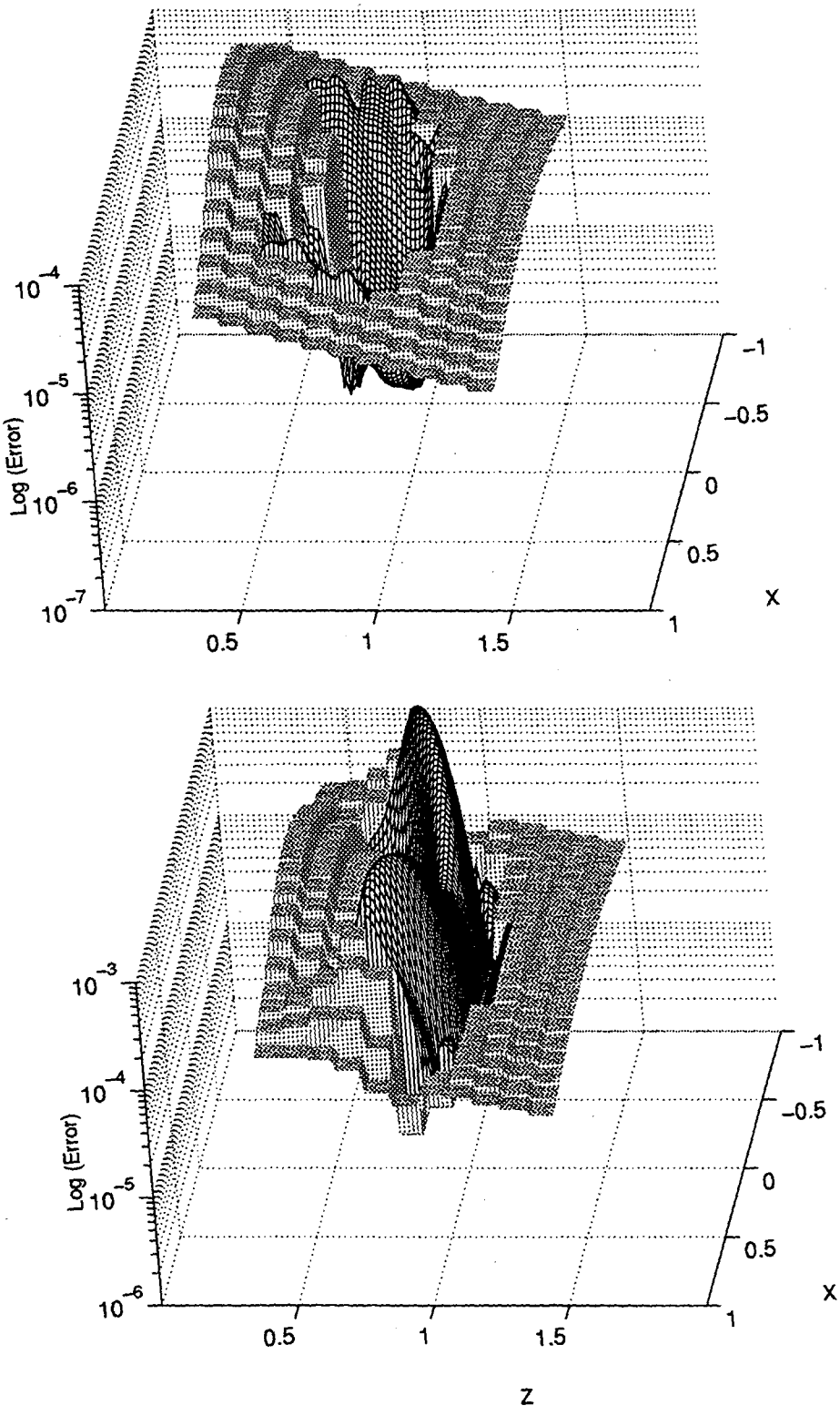


Figure 4.40: Transition to Instability in a Fixed Patch with $\tau\tau = 6$

oscillations appear regardless of the order of temporal interpolation used. Empirically, the critical ratio in time steps has been found to be near twenty.

Thus, with a refinement ratio of four, the inclusion of adaptive time stepping (with jumps around sixteen) does not interfere with the performance on the long-term fixed patch test, as shown in the error plot in Fig–4.41. But with an increased refinement ratio of five (and time step ratios of twenty-five), the method breaks down.

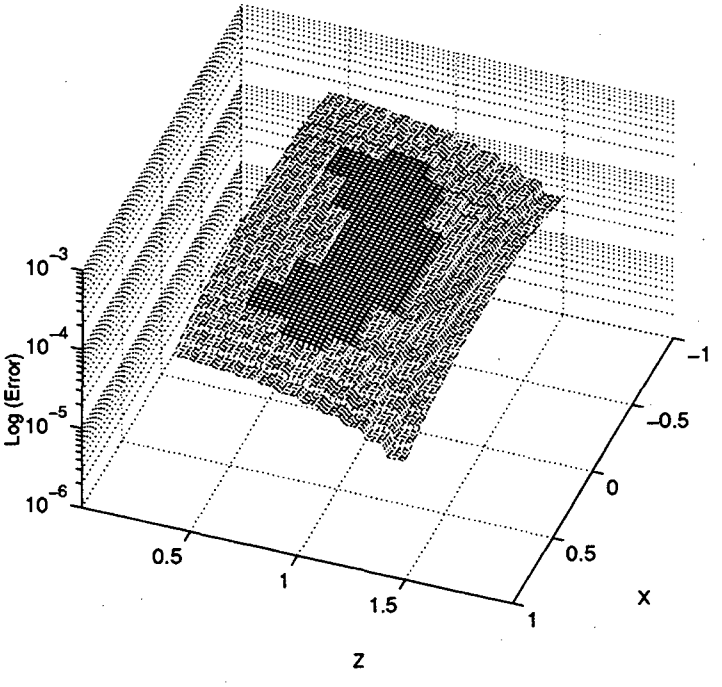


Figure 4.41: Long-Term Stability with Adaptive Time Stepping and $rr = 4$

Since the time step limitation occurs (just) before the spatial one, it needs to be dealt with. The simplest fix is to explicitly restrict the jump in time steps across levels to be less than some critical ratio, $\Delta t_{i-1} \leq \sigma_c \Delta t_i$. This is the route taken in the current implementation.

Chapter 5

Implementation

The foregoing has presented a method for solving the level set equation on an adaptive mesh, but avoided the issue of implementation. There remain serious concerns over the efficiency of the method as a computational tool, primarily regarding the construction of the splines of the previous chapter. This chapter addresses the underlying least squares problems and provides means for combatting their costs.

5.1 Computing the Least Squares Solution

First, a numerical method must be selected for the solution of the weighted least squares problem given by Eq–4.6. A good choice for this purpose (and for the cost-saving devices that will follow) is the direct solution of the normal equations, which, so long as $\mathbf{A}$ has full rank, give the unique minimizer of:

$$\|\omega(\mathbf{Ap} - \varphi)\|_2 = \|(\omega\mathbf{A})\mathbf{p} - (\omega\varphi)\|_2$$

As

$$(\omega\mathbf{A})^{\mathbf{T}}(\omega\mathbf{A})\mathbf{p} = (\omega\mathbf{A})^{\mathbf{T}}\omega\varphi\,. \tag{5.1}$$

Given that $\omega\mathbf{A}$ has full rank (the case when it does not will be addressed momentarily), $(\omega\mathbf{A})^T(\omega\mathbf{A})$ will be symmetric and positive definite and therefore can be factored into a Cholesky decomposition, and the system solved for $\mathbf{p}$ through forward and backward substitutions. The normal equations are a good means of solving the least squares problem so long as $\mathbf{A}^{\mathbf{T}}\mathbf{A}$ is not ill-conditioned (see [18]). For the stencils created as in §4.6, the empirical evidence indicates that the system weighted by inverse distance has a well conditioned

$(\omega\mathbf{A})^T(\omega\mathbf{A})$, with condition numbers on the order of unity, even though $\mathbf{A^TA}$ is often much more poorly conditioned.

The Cholesky factorization will detect degenerate systems for which $\mathbf{A}$ does not have full rank, and for which there is not a unique solution to the least squares problem. In practice, this happens very infrequently. Under the stencil generation algorithm of §4.6, the only degenerate stencil that has been observed is at the corner cell of a 1-nested level, as depicted in the two-dimensional illustration of Fig–5.1. No degeneracies occur in a 2-nested meshes. Without a general guarantee against degeneracies, however, a means of handling them must be incorporated. There are two convenient ways of doing so: the stencil can be expanded to include more cells, or the order of the spline can be reduced. The latter option has been taken here under the assumptions that: (i) it is the extreme sparsity of the local mesh that is causing the degeneracy and (ii) this sparseness also indicates a decreased need for accuracy in the region (see further §5.4).

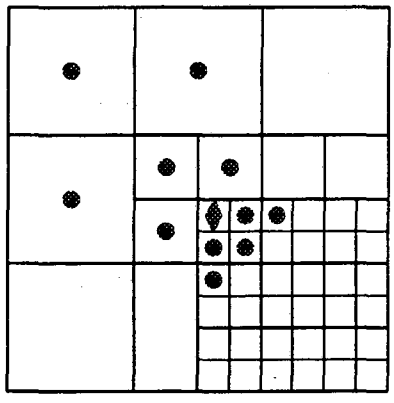


Figure 5.1: A Degenerate Stencil

The essential concern is that of cost. Generating a stencil and constructing the spline as described above for every peripheral cell at every time step would be inordinately expensive. The method would remain $\mathcal{O}(n^2)$, but the constant "in front" would be so huge as to render it useless for anything but the simplest problems. The task at hand is then to reduce this cost, with a clear necessity becoming the retention of information across time steps.

5.2 The Pseudo-Inverse

First of all, the stencils only change when the mesh is changed, so they can be retained throughout the intervals in which the mesh is fixed. In fact, the stencil for a given cell only needs to be replaced when the cells it references are affected by a change to the mesh (which will typically only effect a very small fraction of all cells). Applying this observation all but eliminates the costs associated with generating the stencils.

A similar reduction can be made in the cost of constructing the splines by first rewriting Eq–5.1 as

$$\mathbf{p} = (\omega\mathbf{A}^{\mathbf{T}}\omega\mathbf{A})^{-1}\omega\mathbf{A}^{\mathbf{T}}\omega\varphi = \mathbf{S}\varphi\,,$$

and calling $\mathbf{S} = (\omega\mathbf{A}^{\mathbf{T}}\omega\mathbf{A})^{-1}\omega\mathbf{A}^{\mathbf{T}}\omega$ the pseudo-inverse for the weighted least squares problem. Then since $\mathbf{S}$ remains fixed for a given stencil, it can be pre-computed when the stencil is generated and used to produce splines from φ vectors via the matrix–vector multiply throughout the stencil's existence.

In fact, since only the first and second order terms of the polynomial spline are used (in the production of the first and second derivatives of φ), the entire matrix–vector need not be carried out. The nine rows of $\mathbf{p}$ corresponding to the third order polynomial terms (in three dimensions) can be disregarded and only the three first order and six second order rows computed. The cost of evaluating the derivative at a splined cell is thus $9k$ multiplications and $9k$ additions where k is the number of cells in its stencil. With a typical stencil size being thirty cells, this is still an expensive operation, coming in at about ten times the cost of the standard central difference corollary on a uniform mesh. In §5.4 the topic of this expense will be revisited, but there is a more pressing issue to be addressed first.

Since a large number of peripheral cells can be expected, it would still be computationally expensive, and probably unviable in terms of memory consumption, to compute and store an individual pseudo-inverse operator, $\mathbf{S}$, for each such cell. Many of these cells, however, will have similar stencils and be able share to pseudo-inverse operators. The next section describes how the similarity between stencils can be recognized and exploited to that end.

5.3 Eliminating Symmetries

Many cells will, in fact, share identical stencils, in the sense that their coordinate matrices, the **A**'s of Eq–4.6, will be identical, at least up to a permutation. The possibility of the permutation can be eliminated if the cells of the stencils are sorted by an ordering on their relative positions, the (x_i, y_i, z_i)'s, before the matrices are constructed. Then, instead of building the pseudo-inverse for each peripheral cell, one can be built for each **A** and kept in a look-up table for each identical stencil to reference.

But it is also true that many cells on differing levels will have stencils that vary only by scale, and that still more cells will have stencils that differ only in orientation; see Fig–5.2. The plethora of stencil types (and hence operators) can be be greatly reduced if these symmetries are first eliminated.

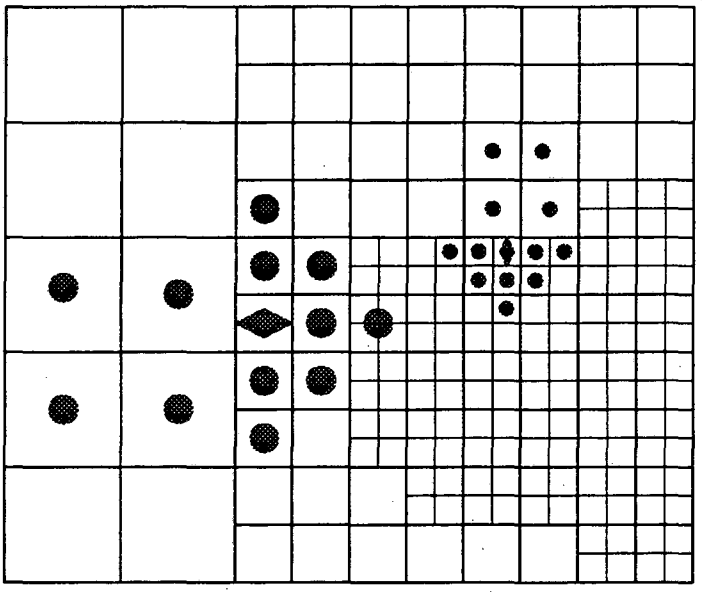


Figure 5.2: Similar Stencils

The scaling symmetry can be dispensed of handily by incorporating the dimensions of a cell, $(\Delta x, \Delta y, \Delta z)$, as length scales for its spline. This is accomplished by creating a diagonal scaling matrix, **H**, and rewriting the polynomial evaluation, **Ap**, as $(\mathbf{AH}^{-1})(\mathbf{Hp})$;

H is chosen so that Eq–4.6 becomes:

$$\underbrace{\begin{pmatrix} \frac{x_1}{\Delta x} & \frac{y_1}{\Delta y} & \frac{z_1}{\Delta z} & \cdots & \frac{x_1 y_1 z_1}{\Delta x \Delta y \Delta z} \\ \frac{x_2}{\Delta x} & \frac{y_2}{\Delta y} & \frac{z_2}{\Delta z} & \cdots & \frac{x_2 y_2 z_2}{\Delta x \Delta y \Delta z} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{x_n}{\Delta x} & \frac{y_n}{\Delta y} & \frac{z_n}{\Delta z} & \cdots & \frac{x_n y_n z_n}{\Delta x \Delta y \Delta z} \end{pmatrix}}_{\mathbf{AH^{-1}}} \underbrace{\begin{pmatrix} (\Delta x) b_1 \\ (\Delta y) b_2 \\ \vdots \\ (\Delta x \Delta y \Delta z) d_{123} \end{pmatrix}}_{\mathbf{Hp}} = \underbrace{\begin{pmatrix} \Delta\varphi_1 \\ \Delta\varphi_2 \\ \vdots \\ \Delta\varphi_n \end{pmatrix}}_{\varphi}$$

The rescaled coordinate matrices (the $\mathbf{AH^{-1}}$'s) will then be invariant of the level for which they are created. Cells from differing levels can thus share pseudo-inverse operators and divide out the appropriate length scales when the spline is formally differentiated. (Something very similar occurs with standard finite difference formulations; it is just restated here in the context of this more elaborate machinery).

Eliminating the dependence of the spline construction on the orientation of the stencil proves to be more difficult. Orientation, here, refers to an underlying symmetry group of reflections and discrete rotations. The elements of this group are linear transformations, which will eventually be incorporated into Eq–4.6 in a style similar to that of the scaling transformations before them; first, an explicit description of the group is needed. Each element of the group corresponds to an orthonormal transformation $\mathbf{Q}$, on $\mathbb{R}^3$, which maps the coordinate axes onto themselves. The set of such transformations form a group of order forty-eight called the group of symmetries of the cube. The elements of G can be denoted as $\mathbf{Q}_{(\mathbf{d_1},\mathbf{d_2},\mathbf{d_3})}$ where each $\mathbf{d_i} \in \{\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}, -\hat{\mathbf{x}}, -\hat{\mathbf{y}}, -\hat{\mathbf{z}}\}$, and $\{\mathbf{d_1}, \mathbf{d_2}, \mathbf{d_3}\}$ is a basis. Then $\mathbf{Q}_{(\mathbf{d_1},\mathbf{d_2},\mathbf{d_3})}$ is defined as being the linear operator which maps $\hat{\mathbf{x}} \mapsto \mathbf{d_1}$, $\hat{\mathbf{y}} \mapsto \mathbf{d_2}$, and $\hat{\mathbf{z}} \mapsto \mathbf{d_3}$. Thus, the columns of $\mathbf{Q}_{(\mathbf{d_1},\mathbf{d_2},\mathbf{d_3})}$ are the $\mathbf{d}$'s: $q_{ij} = (\mathbf{d_j})_i$.

The various elements of G and the subgroups which they generate can also be given a geometrical interpretation as rotations and reflections. For example, rotation (by $\frac{\pi}{2}$) about the z-axis is given by $\mathbf{Q}_{(\hat{\mathbf{y}},-\hat{\mathbf{x}},\hat{\mathbf{z}})}$ and it generates a subgroup of four elements. The geometrical properties of the group make it the proper vehicle for the elimination of the orientational variations of stencils such as the one illustrated in Fig–5.2. One set of generators of G and which will be used in the context of identifying stencils is:

$$\{\, \mathbf{Q}_{(-\hat{\mathbf{x}},\hat{\mathbf{y}},\hat{\mathbf{z}})} \equiv \mathbf{M_x}\,,\ \mathbf{Q}_{(\hat{\mathbf{x}},-\hat{\mathbf{y}},\hat{\mathbf{z}})} \equiv \mathbf{M_y}\,,\ \mathbf{Q}_{(\hat{\mathbf{x}},\hat{\mathbf{y}},-\hat{\mathbf{z}})} \equiv \mathbf{M_z}\,,\ \mathbf{Q}_{(\hat{\mathbf{y}},\hat{\mathbf{x}},\hat{\mathbf{z}})} \equiv \mathbf{M_{xy}}\,,\ \mathbf{Q}_{(\hat{\mathbf{z}},\hat{\mathbf{x}},\hat{\mathbf{y}})} \equiv \mathbf{R}\,\},$$

the first four elements being reflection about the planes $x = 0$, $y = 0$, $z = 0$, and $x = y$ (each of which generate a subgroup of two elements), and the last being a $\frac{2\pi}{3}$ rotation about

the diagonal $x = y = z$ (it generating a subgroup of three elements). In fact, any element of G can be written as

$$\mathbf{Q} = \mathbf{R}^{\beta}\mathbf{M_{xy}}^{\alpha_4}\mathbf{M_z}^{\alpha_3}\mathbf{M_y}^{\alpha_2}\mathbf{M_x}^{\alpha_1}\,, \tag{5.2}$$

with $\alpha_i \in \{0,1\}$ and $\beta \in \{0,1,2\}$.

The next step is to define the appropriate action of G on the set of all stencils. Up to this point, stencils have been described as sets of cells, S', but they can equally well be considered as sets of vectors:

$$S = \{\mathbf{X}(C') - \mathbf{X}(C) :\ C' \in S'\}\,,$$

which (together with a corresponding lists of φ values) contain all of the information needed to construct the splines. Moreover, by taking $\mathbf{Q}S$ to be the image of S under $\mathbf{Q}$, the desired action of G is established. Two stencils S and $\tilde{S}$ are then said to be similar if they lie in the same equivalence class among those given by the orbits of G:

$$S \sim \tilde{S} \quad \Leftrightarrow \quad S = \mathbf{Q}\tilde{S} \quad \text{for some } \mathbf{Q} \in G\,.$$

A family of transforms $\mathbf{Q}_S$ will be called canonical transforms if

$$S \sim \tilde{S} \quad \Rightarrow \quad \mathbf{Q}_S S = \mathbf{Q}_{\tilde{S}}\tilde{S}\,.$$

That is, any two similar stencils must map into identical orientations (called the canonical orientation of their orbit) under the application of their respective canonical transforms. The goal is now to find an algorithm for the generation of canonical transforms.

That the vectors composing S are not arbitrary and have as their origin a stencil of cells provides a very nice shortcut for constructing a canonical transform. In particular, S' must contain a cell $\check{C}$ which is from a coarser level than C's — say level L_{i-k} where $C \in L_i$ and $k > 0$. Then the relative position of $\check{C}$:

$$\mathbf{x} \;=\; \mathbf{X}(\check{C}) - \mathbf{X}(C)$$

can be written as

$$\mathbf{x} \;=\; (m_1\Delta x_{i-k}\,,\, m_2\Delta y_{i-k}\,,\, m_3\Delta z_{i-k}\,) \;+\; \mathbf{v}\,,$$

where the m_i's are integers and $\mathbf{v}$ is the offset of C from its ancestor on L_{i-k}:

$$\mathbf{v} \;=\; \mathbf{X}\left(\mathbf{Parent}^k(C)\right) \;-\; \mathbf{X}(C)\,. \tag{5.3}$$

The canonical transform is nearly determined by $\mathbf{v}$ since this vector will have to be rotated into a canonical orientation in order for $\mathbf{x}$ (and the vectors to all cells from L_{i-k}) to be. (Such an argument cannot be applied directly to the elements in the vector representation of S as applying $\mathbf{Q}S$ generally just permutes that set — the offset $\mathbf{v}$, however, perseveres as $\mathbf{Qv}$.)

The following procedure constructs a canonical transformation for (a set containing) a single vector $\mathbf{v}$. The transformation is built up in the style of Eq–5.2.

$\mathbf{Q_v} = \mathbf{Q}_{\text{vec}}(\mathbf{v})$:

1. $\alpha_i, \beta = 0$
2. If $v_1 < 0$: $\mathbf{v} = \mathbf{M_x v}$, $\alpha_1 = 1$
3. If $v_2 < 0$: $\mathbf{v} = \mathbf{M_y v}$, $\alpha_2 = 1$
4. If $v_3 < 0$: $\mathbf{v} = \mathbf{M_z v}$, $\alpha_3 = 1$
5. While $v_3 < \max(v_1, v_2)$:
   $\mathbf{v} = \mathbf{R v}$, $\beta = \beta + 1$
6. If $v_2 < v_1$: $\mathbf{v} = \mathbf{M_{xy} v}$, $\alpha_4 = 1$,
7. Return $(\mathbf{Q_v} = \mathbf{M_{xy}}^{\alpha_4}\mathbf{R}^{\beta}\mathbf{M_z}^{\alpha_3}\mathbf{M_y}^{\alpha_2}\mathbf{M_x}^{\alpha_1})$

Following this procedure, $\mathbf{v} \mapsto (\mathbf{Q_v})\mathbf{v} \equiv \tilde{\mathbf{v}}$ has the effect of mapping all vectors into the volume:

$$\Omega = \{(x, y, z) \in \mathbf{R}^3 : x \geq 0, \quad y \geq 0, \quad z \geq 0, \quad z \geq y \geq x\}.$$

Its effect on the unit cube (centered about the origin) is shown in Fig–5.3. It should be noted that the vectors on the faces of Ω map to themselves under some non-trivial subgroup of G; those on the interior are fixed only under $\mathbf{I}$. (This subgroup, $H_{\mathbf{v}} = \{\mathbf{Q} \in G : \mathbf{Q_v}\tilde{\mathbf{v}} = \tilde{\mathbf{v}}\}$, is called the subgroup fixing $\tilde{\mathbf{v}}$, and $H_{\mathbf{v}} = \langle I \rangle \Leftrightarrow \tilde{\mathbf{v}} \in \Omega \backslash \partial\Omega$.) In other words, the canonical transformation is fully determined for a given vector if (and only if) it maps the vector to the interior of Ω.

Applying this procedure to the parental offset vector $\mathbf{v}$, as given by Eq–5.3 with $k = 1$ (making the almost always true assumption that $S \cap L_{i-1} \neq \emptyset$), significant progress toward finding a canonical transform for S can be made. In fact, it can be fully determined

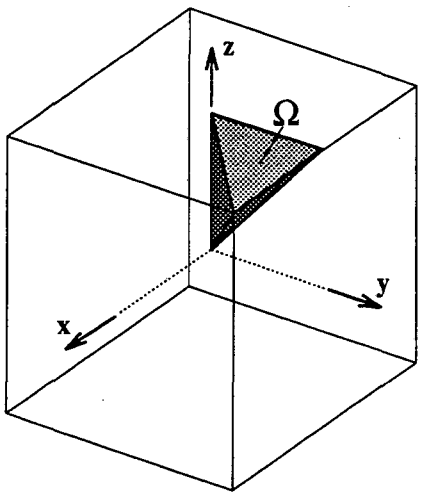


Figure 5.3: Canonical Vector Positions

by $\mathbf{Q_v}$ if $\tilde{\mathbf{v}}$ is in the interior of Ω, since its full determination by a single vector forces its determination for the entire stencil. In the event that this vector is mapped onto the boundary of Ω the contribution to $\mathbf{Q}_S$ from the subgroup $H_\mathbf{v}$ will have to be determined by further investigation of S. In either case, the contribution $\mathbf{Q_v}$ established by the offset to the parent can be precomputed and kept in a brief look up table (of size rr^3) since it only depends on which child of its parent the given cell is.

For the case in which $\tilde{\mathbf{v}}$ lies on a face of Ω, the undetermined symmetry subgroup $H_\mathbf{v}$ can be kept in a corresponding look up table. (The generators of $H_\mathbf{v}$ can be established in terms of the faces on which $\tilde{\mathbf{v}}$ lies.) It is significantly advantageous to eliminate the remaining symmetry, since the number of operators that have to be computed and stored increases proportionally with the size of the unresolved group. For example, with $rr = 2$, the $\mathbf{v}$ for every child cell lies on a diagonal, $\mathbf{v} = (\pm 1, \pm 1, \pm 1)$, and $\mathbf{Q_v}$ will only eliminate a symmetry subgroup of order eight (given by $\langle \mathbf{M_x}, \mathbf{M_y}, \mathbf{M_z} \rangle$) and leave undetermined the contribution to $\mathbf{Q}_S$ from the subgroup $H_\mathbf{v}$ (of order six) given by $\langle \mathbf{M_{xy}}, \mathbf{R} \rangle$. So, for $rr = 2$, if $\mathbf{Q_v}$ is not augmented by a transform from $H_\mathbf{v}$, the number of operators will increase by a factor of six.

Local information about C (such as $\mathbf{v}$, above) is necessarily invariant under the remaining subgroup, so the global effects on S must be looked at instead. To be useful, these effects must also be discerned relatively quickly. One measure on S that is both easy to compute and quite good at distinguishing the stencils generated by the method of §4.6

is the count of the number of cells from $L_{i-k} \cap S$ that lie in a given half-space, such as:

$$n_{x+}(S) = |\{\check{C} \in L_{i-k} \cap S : \mathbf{X}_1(\check{C}) - \mathbf{X}_1(C) > 0\}| .$$

Using these counts, it is possible to construct an algorithm similar to the one for $\mathbf{Q_v}$ above which (usually) finds the contributor to the canonical transform remaining in $H_\mathbf{v}$:

$\mathbf{Q}_{S,H} = \mathbf{Q}_{\text{sten}}(S, H)$:

1. $\alpha_i, \beta = 0$
2. If $\mathbf{M_x} \in H$:
   If $n_{x+}(S) < n_{x-}(S)$: $S = \mathbf{M_x}S$, $\alpha_1 = 1$
3. If $\mathbf{M_y} \in H$:
   If $n_{y+}(S) < n_{y-}(S)$: $S = \mathbf{M_y}S$, $\alpha_2 = 1$
4. If $\mathbf{M_z} \in H$:
   If $n_{z+}(S) < n_{z-}(S)$: $S = \mathbf{M_z}S$, $\alpha_3 = 1$
5. If $\mathbf{R} \in H$:
   While $n_{z+}(S) < \max(n_{y+}(S), n_{y+}(S))$:
   $S = \mathbf{R}S$, $\beta = \beta + 1$
6. If $\mathbf{M_{yz}} \in H$:
   If $n_{z+}S > n_{y+}S$: $S = \mathbf{M_{yz}}S$, $\alpha_4 = 1$
7. If $\mathbf{M_{xy}} \in H$:
   If $n_{y+}(S) < n_{x+}(S)$: $S = \mathbf{M_{xy}}S$, $\alpha_5 = 1$
8. Return $(\mathbf{Q}_{S,H} = \mathbf{M_{xy}}^{\alpha_5}\mathbf{M_{yz}}^{\alpha_4}\mathbf{R}^{\beta}\mathbf{M_z}^{\alpha_3}\mathbf{M_y}^{\alpha_2}\mathbf{M_x}^{\alpha_1})$

The real algorithm does a little more work in the case of equality on the various tests, but the above gives the basic idea. The canonical transform for S is then taken to be:

$$\mathbf{Q}_S = \mathbf{Q}_{S,H_\mathbf{v}}\mathbf{Q_v} ,$$

which now needs to be applied in Eq–4.6.

Much as for the scaling symmetry, the polynomial evaluation $\mathbf{Ap}$ will be re-written as as $(\mathbf{A}\bar{\mathbf{Q}}^{\mathrm{T}})(\bar{\mathbf{Q}}\mathbf{p})$, where $\bar{\mathbf{Q}}$ is built from $\mathbf{Q}_S$. Now $\mathbf{Q}_S$ applies to the coordinates of the stencil, the (x_i, y_i, z_i)'s, and from the definition of the symmetry group it can be written

as $\mathbf{Q} = \mathbf{PD}$ where $\mathbf{P}$ is a permutation and $\mathbf{D}$ is diagonal with $d_{ii} \in \{1, -1\}$. In terms of its application to $\mathbf{A}$ and $\mathbf{p}$, it follows that $(\mathbf{A}\bar{\mathbf{Q}}^{\mathrm{T}})$ and $(\bar{\mathbf{Q}}\mathbf{p})$ will be corresponding column and row permutations (and negations) given by $(\mathbf{A}\bar{\mathbf{P}}^{-1}\bar{\mathbf{D}}^{-1})$ and $(\bar{\mathbf{P}}\bar{\mathbf{D}}\mathbf{p})$ for an appropriate choice of $\bar{\mathbf{P}}$ and $\bar{\mathbf{D}}$. The "transformed spline," $\mathbf{p}' = \bar{\mathbf{Q}}\mathbf{p}$, can then be computed via the pseudo-inverse of $(\mathbf{A}\bar{\mathbf{Q}}^{\mathrm{T}})$, and from there it is easy to reconstruct the desired spline via $\mathbf{p} = \bar{\mathbf{D}}^{-1}\bar{\mathbf{P}}^{-1}\mathbf{p}'$.

The entire transformation scheme for the stencil and spline construction can now be summarized as:

$$\underbrace{(\mathbf{PAH}^{-1}\mathbf{Q}^{\mathrm{T}})}_{\mathbf{A}'}\underbrace{(\mathbf{QHp})}_{\mathbf{p}'} = \underbrace{(\mathbf{P}\varphi)}_{\varphi'} . \tag{5.4}$$

The reordering of φ by $\mathbf{P}$ is accomplished by a sort on the list of cells in the stencil, and this is take care of once and for all when the stencil is first built. Also, when the stencil is built, the pseudo-inverse of $\mathbf{A}'$ is looked up in (and perhaps created for) a hash table of operators for the canonical stencils . To compute the spline $\mathbf{p}$, the transform spline $\mathbf{p}'$ is first computed by the partial matrix–vector multiply described in §5.2 and then scaled and permuted into $\mathbf{p}$. The only significant cost then becomes the matrix–vector multiply.

5.4 Focusing Effort

The cost has been successfully reduced to that of the matrix–vector multiply, but it is still high. It would also seem to be an integrable part of the method and therefore unavoidable, but that is not entirely true. For specific applications, it is possible to selectively choose where the accuracy of the spline construction of Eq–4.6 and Eq–5.4 is truly needed.

For the level set method, the evidence from Chapter 4 would seem to indicate that the use of the method is only essential when a transition in refinement lies along a level surface. The critical region is that in which the zero level surface passes from one level of refinement into another. The theory of §1.3 adds weight to this claim, in that inaccuracy away from the level surface should not affect its motion. The goal here is to find a reduced set of cells that require the full splined treatment.

It becomes a straight forward operation to determine which peripheral cells need the added attention if the refinement criterion for cells are broken into two categories: those which apply because of the proximity to the zero level surface and those which apply because of any other factor (e.g. because of locally high curvature). Then any cell which finds itself

to be on the periphery because of a change in a criterion in the latter category should be treated with the full machinery of Eq–4.6. For a refinement strategy in which the levels are, generally, bands about the zero level surface, with some transitions in the level of banding along the surface, the number of such cells will be $\mathcal{O}(n)$. Therefore, a great savings will be realized if the full spline construction only has to be applied in these cells.

And, indeed, this turns out to be the case — for the level set equation, the evidence here will indicate that even interpolation can be used along the peripheral bands about the surface (even though all the evidence indicates that interpolation is a very bad idea for the heat equation). Returning to the problem of §4.7 in which the patch of refinement was restricted to be within a cone about the z-axis, the cells selected for the special treatment will be all those lying along the surface of the cone. The spline-based method is then used for these cells, while the peripheral cells on the two faces parallel to the level surfaces of φ are updated by linear interpolation from the coarse cells.

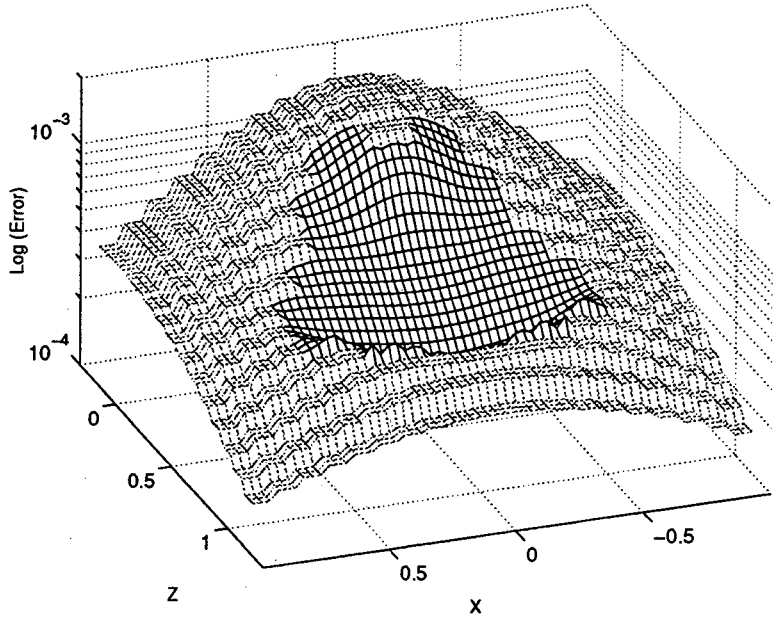


Figure 5.4: Error under Patched Refinement and Mixed Methods

The resultant scheme is successful in that it maintains an error comparable to that of the fully splined method along the zero level surface. The error is plotted at $t = 0.3$ in Fig–5.4, and the increase in error due to the partial reliance on interpolation is shown in

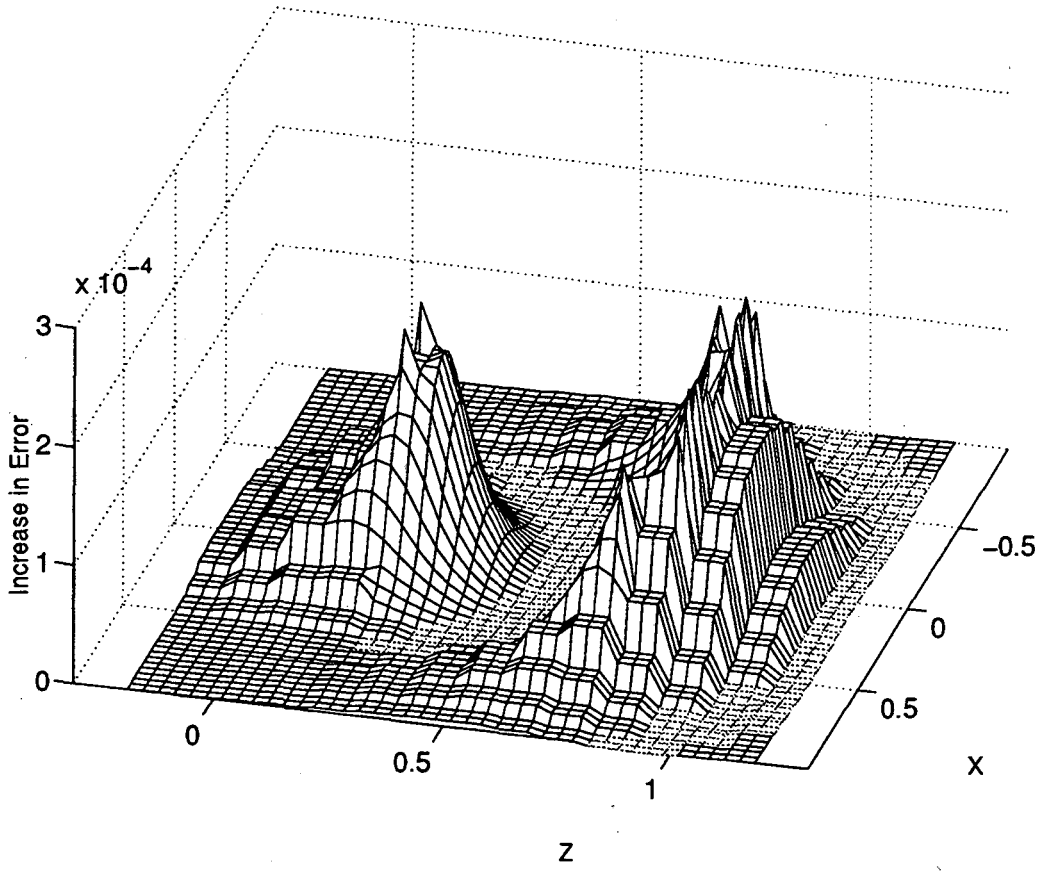


Figure 5.5: Comparison of Mixed Methods to Fully Splined Method

Fig–5.5. The error is seen to suffer away from the zero level surface, but the solution is not degraded on the zero level surface. In fact, for this particular problem, it is somewhat improved along the center of the patch, the grey portions of the plot in Fig–5.5 indicating a decrease in error.

With the success of this simplification, a very efficient and accurate method is provided for tracking surface evolutions. The restriction of the spline-based construction to a limited subset of the peripheral cells eliminates the associated cost in the asymptotic scaling (or more explicitly, for $n \gg 10$). The method therefore scales just as does the uniform method in terms of the cost per number of data points, with a slight (machine dependent) overhead incurred because of the indirect referencing required by the data structure.

The use of interpolation away from the zero level set also has a secondary effect on the cost. Because the accuracy is degraded near the edges of the band of refinement (as

in Fig–5.5), the minimal width of these bands is somewhat increased over the one required when splines are used everywhere. The cost will increase proportionally with the number of cells that have to be added. The net effect, however, is still a considerable savings over the cost of employing the spline based method everywhere on the periphery.

The next chapter briefly discusses an entirely different approach for updating the peripheral cells. This approach offers a substantially reduced cost from that of the spline-based method, while still maintaining the accuracy — for the two-dimensional model problem, at least. It is not yet clear whether or not this second method can be successfully generalized to higher dimensions. For the level set equation, the possible savings from the second technique are minimal, since the observations in this section have shown that the cost of splining can largely be avoided. For parabolic problems in general, however, the savings can be substantial.

Chapter 6

A Second Technique

Altas and Stephenson mention a technique in [3] which can be adapted here. It is restricted in its application, however, because of the involvement of a trick that only applies to refinement ratios of two and only to a different style of refining the mesh than the one discussed in Chapter 3. Before introducing the technique, it is therefore necessary to outline this other style of refinement.

6.1 Point-Centered Adaptive Meshes

The style of mesh described in Chapter 3 is called a cell-centered mesh, since values are stored at the centers of cells, and these cells are subdivided in the refinement. The alternative described here is based on "grid points" and hence called a point-centered mesh. Most of the terminology of Chapter 3 can be carried over (with somewhat distorted definitions). In particular, the mesh will still be described by a refinement ratio, rr, which in the context of this technique will be restricted to $rr = 2$.

The fundamental object will become a point, P, instead of a cell; $\mathbf{X(P)}$ will now give its position, and $(\Delta x_P, \Delta y_P, \Delta z_P)$ will become something called the "grid spacing". The main difference is that the points will have 3^n children (for $rr = 2$), with the positions of the children of P given (in three dimensions) by

$$\mathbf{X}(P) + \left(m_1 \frac{\Delta x_P}{2}, m_2 \frac{\Delta y_P}{2}, m_3 \frac{\Delta z_P}{2} \right) \qquad m_i \in \{-1, 0, 1\}.$$

An example of such a mesh in two dimensions appears in Fig–6.1 and should serve to clarify the way in which the definition of the mesh has been altered. The circles indicate

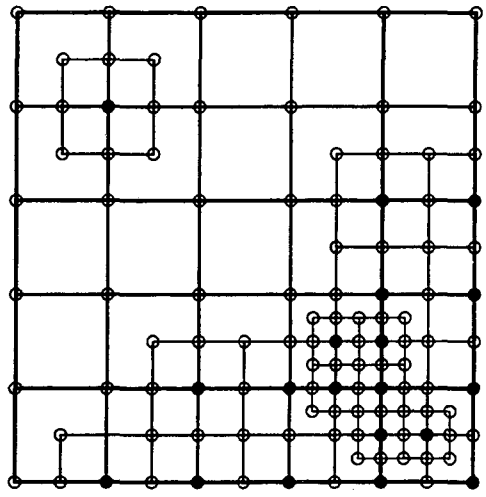


Figure 6.1: A Point-Centered Mesh

the "grid points" and the lines connecting them, the "grid;" the points with children have been indicated by filled circles. In a point-centered mesh, every parent is coincident with one of its children (as is the case for cell-centered meshes with odd refinement ratios — in fact cell-centered and point-centered meshes with odd refinement ratios are identical). For even refinement ratios, there can be coincident points on the same level which must be identified together as a single point. As a result, some points of the mesh can have multiple parents, or put otherwise, points can share children.

The fact that was noted about the peripheral points in [3] is that they are all surrounded by four uniformly spaced points. For "edge" children, these points lie along the coordinate axes; for "corner" children, they are rotated by $\frac{\pi}{4}$. The standard central difference formulas can therefore be used for the first and second derivatives along the x' and y' axes and (if they are not combined into a rotationally invariant form as is often the case) these can then be formally rotated at the corner children.

6.2 The-Two Dimensional Model Problem (Revisited)

Applying this style of adaptive mesh to the model problem of §4.4 yields a mesh like the one depicted in Fig–6.3. For the heat equation, the rotated central differences discussed in the last section can be applied directly since the Laplacian is rotationally invariant and does not involve the mixed derivative u_{xy}. Returning the problematic test case of §4.4

with $k_1 = 1$ and $k_2 = 2$, the method is seen to perform well, as is illustrated by the error graph in Fig–6.2 (being for a coarse mesh of 30×30 points and $t = 0.001$, just as for the first plot of Fig–4.24). Once again, the difficulty along the periphery has been eliminated in this case, and in general, the method performs well under various initial conditions for the heat equation.

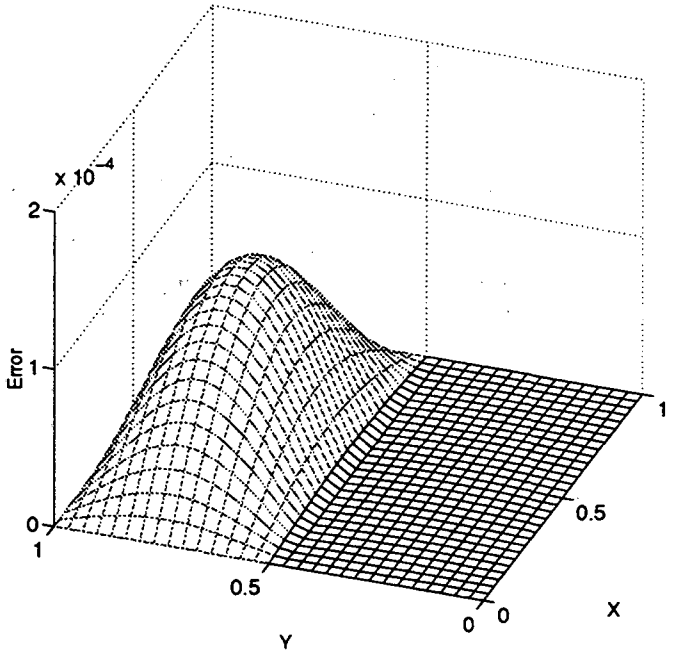


Figure 6.2: Heat Equation with $k_1 = 1$, $k_2 = 2$, Revisited

For the curvature evolution equation (Eq–1.5) the inclusion of the cross derivative presents a problem, since there does not exist a rotated frame for the peripheral cells which provides all of the points needed for the standard central difference approximations. The technique of [3] did not allow cross derivatives for this reason, but it can be extended in the following manner. Fig–6.3 shows two stencils (for the model problem) from which the cross derivative can be approximated by the centered but unproportionately distributed "corner" points. These points have been labeled a–d in the rotated stencil of Fig–6.3. The cross derivative for this stencil can therefore be approximated as

$$\frac{(\varphi_b - \varphi_a) - (\varphi_d - \varphi_c)}{4h^2},$$

where h is the uniform coarse grid spacing, and φ is evaluated at the corners as labeled in the figure.

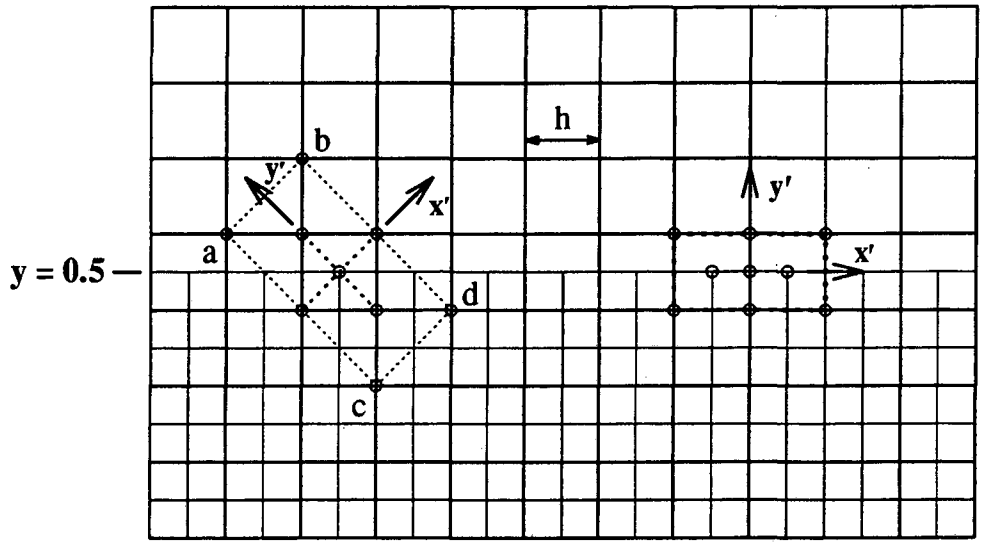


Figure 6.3: Stencils for the Model Problem

The curvature evolution equation is also rotationally invariant, so the difference approximations can be substituted directly, and the resulting method again performs nicely. The errors for the flows in various directions used in Fig–4.23 are re-plotted for this method in Fig–6.4. The results are again nearly identical to the spline based results, with a slight improvements on the order of 10^{-7} as shown in Fig–6.5 for the case of $\theta = \frac{\pi}{8}$.

The success in this collection of test cases provides strong evidence that this method should work for the curvature dependent level set equation on arbitrary two-dimensional, point-centered adaptive meshes (with $rr = 2$). For three-dimensional meshes an additional difficulty arises which will be discussed in §6.4. But first, §6.3 addresses the issue of generalizing the method to arbitrary refinement ratios and to cell-centered meshes.

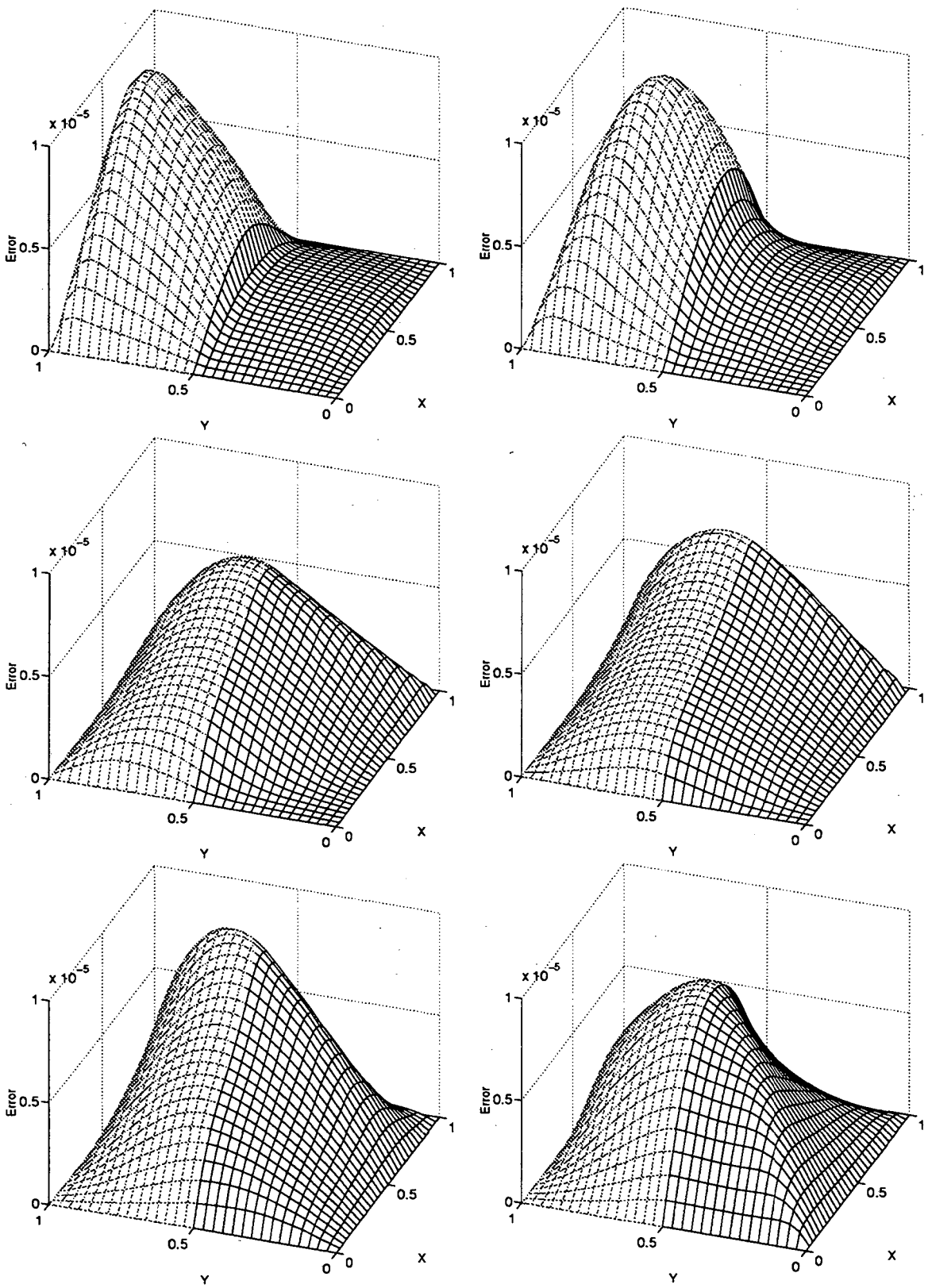


Figure 6.4: Curvature Flow in Various Directions, Revisited

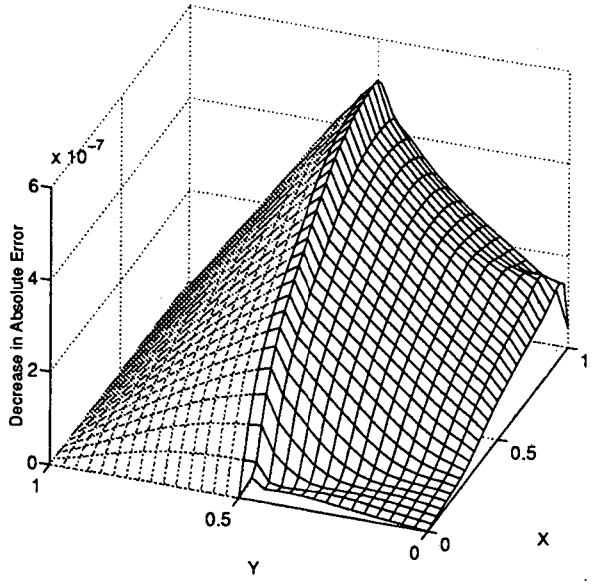


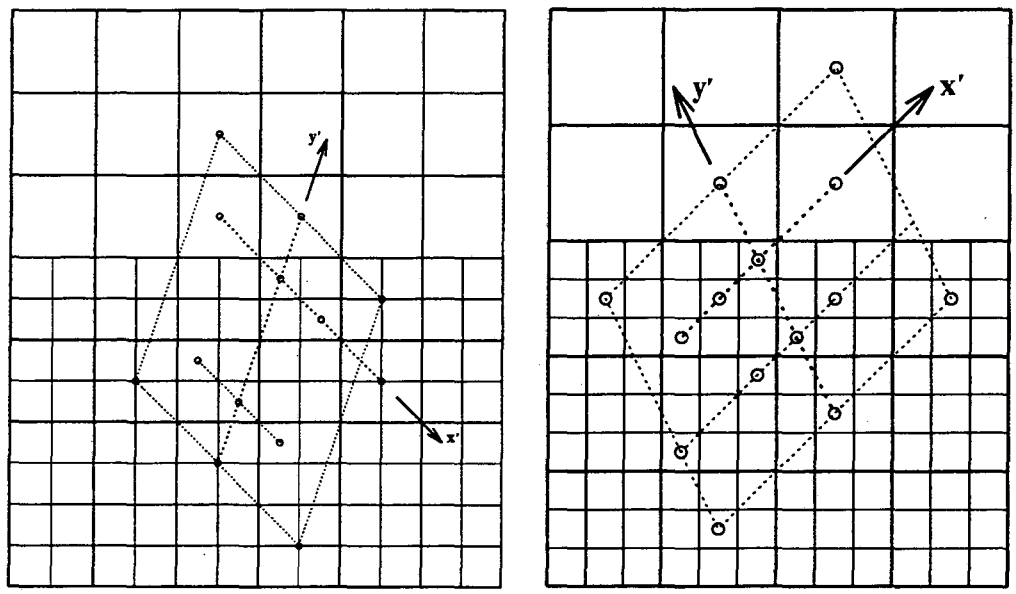
Figure 6.5: Comparison to the Splined Technique for $\theta = \frac{\pi}{8}$

6.3 Attempts at a Generalization

The choice of a point-centered mesh in two dimensions, with $rr = 2$ and that is 2-nested, allows the use of the special centered stencils shown in Fig–6.3. For a higher refinement ratio or for a cell-centered mesh, such stencils do not exist. There are, however, somewhat similar stencils in skewed coordinate systems for these meshes.

Figure Fig–6.6 shows the relevant constructions for $rr = 3$ and for a cell-centered mesh with $rr = 2$.

In each case, a skewed coordinate system $(\mathbf{x}', \mathbf{y}')$ is formed for which the stencil points lie at lattice points. Some hope can be held, then, for a method that evaluates the derivatives with respect to this coordinate system and then transforms them to the to the derivatives with respect to the original $(\mathbf{x}, \mathbf{y})$ coordinates via the appropriate Jacobian. There is also a second difficulty in that the selection of centered stencil points as in §6.2 is impossible in this setting. Various uncentered choices could be made — some candidates have been indicated in the figure — and uncentered finite difference equations, in the style of Eq–4.2, applied at these stencil points. None of the resulting schemes, however, have been found to be successful. Their performance is generally comparable to that of the initial spline based method of §4.4.



(a) Cell-Centered with $rr = 2$ (b) For $rr = 3$

Figure 6.6: Similar Stencils for Different Refinement Styles

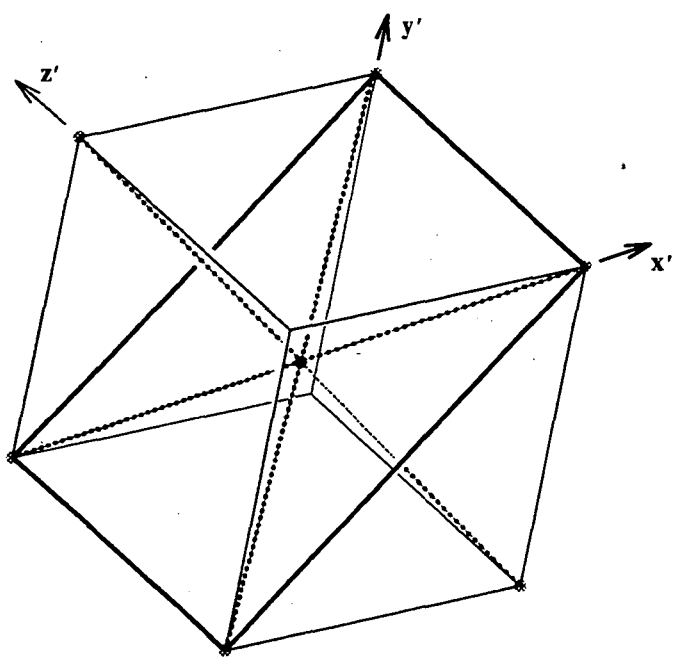


Figure 6.7: Skewed Coordinates in Three Dimensions

6.4 The Three-Dimensional Problem

There is also a difficulty in extending the method to a three-dimensional mesh, even in the point-centered, $rr = 2$ case. The problem is that for corner children, a stencil does not exist in a simply rotated orientation. Instead, a non-orthogonal (skewed) coordinate system as depicted in Fig–6.7 has to be used. Here the non-orthogonality of the $\mathbf{z}'$-axis to the $\mathbf{x}'\mathbf{y}'$ plane and the skewing of the $\mathbf{x}'$ and $\mathbf{y}'$ axes can easily be observed; symmetric relationships hold between all the axes. The situation is similar to the skewed coordinates used in §6.3, and a similar change of variables must to be applied to the computed derivatives. Centered stencil points can be chosen in this coordinate system, however, and that distinction offers reason to believe that the scheme might succeed where those of §6.3 failed.

On the other hand, there is a worrisome asymmetry in the choice of the coordinates in Fig–6.7. One of the diagonals of the cube has been singled out and the coarse data along that axis is not referenced. The asymmetrical favoring of a direction and the lack of a means by which to choose that direction are both reminiscent of the situation in §4.4 when a specified number of stencil points could not be chosen symmetrically. The difficulty that developed there raises significant concern over the three-dimensional method being suggested here. An implementation of the method is currently being developed in order resolve this issue.

Chapter 7

Concluding Remarks

This thesis has introduced a method for solving the three-dimensional level set equation on a fully adaptive mesh. In doing so it has addressed the difficult subsidiary problem of the numerical solution of second order parabolic partial differential equations on such a mesh. The difficulties were analyzed in Chapter 4 and resolved there with a new and very robust, though expensive, method. Chapter 6 extends another, less expensive, method to the solution of two-dimensional parabolic equations on a limited type of adaptive mesh. Should the generalization of that method to three dimensions as outlined in §6.4 prove viable, it would lead to a more efficient algorithm for parabolic equations in three dimensions. Work in that direction is currently underway.

At this juncture, the spline-based method has been shown to be highly accurate under weak conditions on the adaptive mesh. Moreover, its relatively high cost was shown (in §5.4) to be largely irrelevant in the context of the level set equation. The method that has been presented in this thesis therefore provides a very efficient means for the solution of that equation. A forthcoming paper [28] covers its application to a variety of surface evolution problems, and compares its speed and accuracy to both the full (uniform) level set discretization and the narrow band formulation. That paper also analyzes the role of refinement criteria in the applications (and their effects on speed and accuracy as well), which is a topic that has only been briefly touched on here.

In addition, there remain a variety of loose ends and and open problems which could be fruitfully pursued. The method used for the hyperbolic term should be extended to a higher order method and its behavior on the adaptive mesh studied. The failure of the various parabolic schemes leaves a range of unresolved questions, and the several

heuristic arguments given in Chapter 4 could perhaps be made rigorous. Furthermore, such investigations could possibly lead to a proof of the accuracy of the spline-based method, which to this point can only be claimed through empirical observation. And finally, all of the methods presented would benefit greatly in a parallel or vector based implementation.

What has been accomplished is the complete development of the first adaptive method for the numerical solution of the level set equation. The method has been shown to be both efficient and accurate over a range of test problems. The scope of these test problems is such that they give a strong indication that the method will be successful in general application. The further examples given in [28] add weight to this proposition, and hopefully, they will soon be supplemented through application to the modeling of a "real world," physical process.

This work was motivated by the wealth of physical problems in which there is an evolving surface that exhibits a range of relevant length scales. With its completion, there comes an enormous opportunity to study these problems at a more appropriate level of resolution than has previously been possible.

Bibliography

[1] D. Adalsteinsson and J.A. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118(2):269–277, 1995.

[2] D. Adalsteinsson and J.A. Sethian. A unified level set approach to etching, deposition and lithography I: Algorithms and two-dimensional simulations. *Journal of Computational Physics*, 120(1):128–144, 1995.

[3] I. Altas and J. Stephenson. A two-dimensional adaptive mesh generation method. *Journal of Computational Physics*, 94:201–224, 1991.

[4] S. Angenent, D.L. Chopp., and T. Ilmanen. On the singularities of cones evolving by mean curvature. *Communications in Partial Differential Equations*, 1995.

[5] M. Berger. Stability of interfaces with mesh refinement. *Mathematics of Computation*, 45(142):301–318, October 1985.

[6] M. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82(1):62–84, 1989.

[7] M. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53:484–512, 1984.

[8] D.L. Chopp. Computing minimal surfaces via level set curvature flow. *Journal of Computational Physics*, 106:77–91, 1993.

[9] D.L Chopp. Numerical computation of self-similar solutions for mean curvature flow. *Experimental Mathematics*, 3(1):1–15, 1994.

[10] A.J. Chorin. Flame advection and propagation algorithms. *Journal of Computational Physics*, 35:1–11, 1980.

[11] A.J. Chorin. Curvature and Solidification. *Journal of Computational Physics*, 57:472–490, 1985.

[12] Chopp D.L. and J.A Sethian. Flow under curvature: Singularity formation, minimal surfaces and geodesics. *Experimental Mathematics*, 2(4):235–255, 1993.

[13] L.C. Evans, H.M. Soner, and P.E. Souganidis. Phase transitions and generalized motion by mean curvature. *Communications on Pure and Applied Mathematics*, 45:1097–1123, 1992.

[14] L.C. Evans and J. Spruck. Motion of level sets by mean curvature I. *Journal of Differential Geometry*, 33:635, 1991.

[15] L.C. Evans and J. Spruck. Motion of level sets by mean curvature II. *Transactions of the American Mathematical Society*, 330(91), 1992.

[16] L.C. Evans and J. Spruck. Motion of level sets by mean curvature III. *Journal of Geometric Analysis*, 2:121–150, 1992.

[17] L.C. Evans and J. Spruck. Motion of level sets by mean curvature IV. *Journal of Geometric Analysis*, 5(1):77–114, 1995.

[18] G.H. Golub and C.F. Van Loan. *Matrix Computations.* John Hopkins University Press, 1989.

[19] M. Grayson. The heat equation shrinks embedded plane curves to round points. *Journal of Differential Geometry*, 26:285, 1987.

[20] M.E. Gurtin. On the two-phase Stefan Problem with interfacial, energy and entropy. *Archive for Rational Mechanics and Analysis*, 96:199–241, 1986.

[21] M.E. Gurtin. *Thermomechanics of Evolving Phase Boundaries in the Plane.* Oxford University Press, 1993.

[22] J.J. Helmsen. *A Comparison of Three-Dimensional Photolithography Development Methods.* PhD thesis, University of California, Berkeley, 1994.

[23] C.W. Hirt and B.D. Nicholls. Volume of fluid (VOF) method for dynamics of free boundaries. *Journal of Computational Physics*, 39:201–225, 1981.

[24] D.A. Kessler and H. Levine. Stability of dendritic crystals. *Physics Review Letters*, 57:3069–3072, 1986.

[25] W.E. Lorenson and H.E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[26] R. Malladi and J.A. Sethian. Image processing via level set curvature flow. *Proceedings of The National Acadademy of Sciences*, 92(15):7046–7050, 1995.

[27] R. Malladi, J.A. Sethian, and B.C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(2):158–175, February 1995.

[28] B. Milne and J.A. Sethian. Adaptive mesh refinement for level set methods for propagating interfaces. To be submitted for publication, *Journal of Computational Physics*, 1995.

[29] W. Mulder, S.J. Osher, and J.A. Sethian. Computing interface motion in compressible gas dynamics. *Journal of Computational Physics*, 100:209–228, 1992.

[30] W. Noh and P. Woodward. A simple line interface calculation. In *Proceedings of the Fifth International Conference on Fluid Dynamics*. Springer-Verlag, 1976.

[31] S. Osher and J.A. Sethian. Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulation. *Journal of Computational Physics*, 79:12–49, 1988.

[32] E.G. Puckett. A Volume-of-Fluid interface tracking algorithm with applications to computing shock wave refraction. In *Proceedings of the 4th International Symposium on Computational Fluid Dynamics*, Davis, California, 1991.

[33] C. Rhee, L. Talbot, and J.A. Sethian. Dynamical study of a premixed V-flame. *Journal of Fluid Mechanics*, 300:87–115, 1995.

[34] J.A. Sethian. *An Analysis of Flame Propagation.* PhD thesis, University of California, Berkeley, 1982.

[35] J.A. Sethian. Curvature and the evolution of fronts. *Communications in Mathematical Physics*, 101:487–499, 1985.

[36] J.A. Sethian. Numerical algorithms for propagating interfaces: Hamilton-Jacobi equations and conservation laws. *Journal of Differential Geometry*, 31:131–161, 1990.

[37] J.A. Sethian. Curvature flow and entropy conditions applied to grid generation. *Journal of Computational Physics*, 115:440–454, 1994.

[38] J.A. Sethian. Algorithms for tracking interfaces in CFD and material science. *Annual Review of Computational Fluid Mechanics*, 1995.

[39] J.A. Sethian. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Material Science.* Cambridge University Press, 1996.

[40] J.A. Sethian and J.D. Strain. Crystal growth and dendritic solidification. *Journal of Computational Physics*, 98:231–253, 1992.

[41] M. Spivak. *A Comprehensive Introduction to Differential Geometry.* Publish or Perish, 1979.

[42] M. Sussman, P. Smereka, and S.J. Osher. A level set method for computing solutions to incompressible two-phase flow. *Journal of Computational Physics*, 114:146–159, 1994.

Ernest Orlando Lawrence Berkeley National Laboratory
One Cyclotron Road | Berkeley, California 94720

Prepared for the U.S. Department of Energy under Contract No. DE-AC03-76SF00098