THE UNIVERSITY OF READING

Moving Mesh Methods for Non-Linear Parabolic Partial Differential Equations

by

K.W.Blake

Department of Mathematics

This thesis is submitted for the degree of
Doctor of Philosophy

October 2001

Abstract

This thesis is primarily concerned with the use of moving mesh methods for the approximate solution of non-linear partial differential equations of parabolic type. Such methods have become a popular means for the solution of problems which may contain sharp features that are hard to approximate, whilst efficiently managing computational overheads.

Initially, a novel moving grid technique known as *Contour Zoning* is discussed. This 'static' method is able to reduce numerical resources by grouping together sets of nodes as a moving contour of the solution.

Motivated by the findings from this method, a 'dynamic' moving mesh method is developed, initially powered by conservation of mass and later by equidistribution principles. The unusual feature of the method is that the resulting system is solved entirely through the grid co-ordinates, with the underlying partial differential equation solution being constructed algebraically *a posteriori* from the mesh. Numerical results are compared with an analytical solution to the porous media equation which drives the development of the method. Self-similar theory is also used to verify our approximate solutions.

In one dimension the method is highly successful when applied to non-linear diffusion problems incorporating moving or fixed boundaries and problems with blow-up. However, in higher-dimensions we encounter oscillatory solutions emanating from an inability to manipulate our solution technique into square, invertible systems which satisfy all of the conditions required for this style of mesh movement.

Acknowledgements

I would first like to thank my supervisors, Professor Mike Baines and Dr Pete Sweby for their constant goodwill, expertise and enthusiasm, with which I doubt the last three years would have been as rewarding. My thanks extend to other members of staff and fellow postgrads in the maths department both past and present.

I owe special thanks to my family for their love, support and encouragement during my entire time at university.

I would also like to thank all my friends both at Reading and at home. Special thanks to Katie, Wako & Kev, Emma, Ross, Caroline, Oli and Marnie.

Lastly, I am grateful to the University of Reading for providing the Paul White studentship to support this work.

Declaration

I confirm that this is my own work and the use of all material from other sources has been properly and fully acknowledged.

Contents

Chapter 1

Introduction

The use of adapted meshes in the numerical solution of partial differential equations (PDEs) has become a popular technique for improving existing approximation schemes. In problems in which features with large solution variations are common, such as steep fronts and sharp variations, the choice of a non-uniform mesh can not only retain the accuracy but also improve the efficiency of an existing method by concentrating mesh points within regions of interest. This thesis is primarily concerned with the use of such methods for the solution of non-linear parabolic PDEs of the form

$$u_t = (D(u)u_x)_x + Q(u).$$

In particular, we shall be considering problems involving non-linear diffusion and solution blow-up.

The advantages of such an approach go hand in hand. Firstly, since such areas of interest in the mesh inevitably involve large variations in the solution, for any numerical scheme a smaller spatial resolution in the mesh is essential for a reliable approximation and accurate representation. However, to enforce this requirement over the entire grid will be an expensive process, especially in higher dimensions. It becomes obvious then, that for efficiency it is desirable to concentrate nodes and hence computational effort in those parts of the grid that require most attention. A successful approach will then ensure suitable mesh resolution whilst retaining computational efficiency.

In general there are three classifications of grid adaption. The first, h-refinement, adds extra nodes to an existing mesh to improve local grid resolution. A second technique, p-refinement, employs higher order numerical schemes to improve local accuracy as well as to approximate troublesome derivatives. The third approach is r-refinement, which maintains the existing number of nodes globally but relocates them strategically and, more importantly, efficiently over the domain. It is this latter idea that we shall be concerned with in the course of this thesis.

Having decided upon the use of an adapted mesh for the problem in hand the obvious question is, how does one choose a suitable grid? There are many techniques reported in the literature relating to both fixed and moving grids. When considering a moving mesh algorithm for the solution of a time-dependent PDE, the techniques which underpin the grid movement are often found in the literature for the generation of adapted grids for the numerical approximation to steady problems. One such technique is *equidistribution*, first introduced by de Boor [23], involving locating mesh points such that some measure of the solution geometry or error is equalized over each subinterval. Another technique is functional minimisation, which generates the mesh from a variational principle. The line of thinking is that if the properties of a mesh can be measured they can also be controlled. Early examples of this methodology are found in the work of Winslow [77] and Brackbill & Saltzman [13], who adapt the mesh so that it exhibits desirable properties such as smoothness and orthogonality. These techniques can be carried over into a moving mesh method, the difference being that the mesh is moved in conjunction with the developing numerical approximation through time in such a way that the underlying motivations of the gridding strategy are retained. It is at this point where two categories of r-refinement appear, static and dynamic methods.

For static methods, the approximate solution is defined initially on a given mesh. During the calculation a new mesh (which may or may not have the same number of nodes) is generated using an existing grid generation technique. The solution is then interpolated onto the new mesh, so the redistribution and/or addition of grid points and the interpolation are all carried out at a fixed time in the solution process. Although successful, these methods carry large computational overheads

due to the intermittent changes in the data structure describing the mesh in any arithmetic code ([65],[55],[61]).

In dynamic methods, a mesh equation is employed to prescribe the speeds of nodes in order to move a mesh in such a way that gridpoints remain concentrated in regions of rapid variation as the solution evolves with time. In general in these methods the number of nodes in the grid remain the same. For this type of moving mesh method two coupled equations need to be considered, the moving mesh equation controlling the development of the mesh and that associated with the underlying problem ([44], [43], [59]). The mesh then develops continuously with no interpolation steps required. During this thesis we shall look at both types of moving mesh method, with the principle interest lying in the former.

Both static and dynamic methods require some underlying motivation for their distributive strategies. It seems that in general the principles behind moving a grid through time employ the same techniques involved in generating a stationary mesh, for example a static method may redistribute grid points via an equidistribution principle. Indeed, many dynamic methods are derived by introducing a time derivative into an existing grid generation technique. For instance, in the functional minimisation approach to grid generation moving mesh equations can be derived from solving the gradient flow equations associated with the minimal grid functional (see Huang and Russell [47], [48]).

This thesis begins by outlining some of the existing literature on both grid generation and moving mesh methods. The next chapter will introduce both the equidistribution and functional minimisation approaches to grid generation. It is hoped that the reader, by first understanding the aims and means of grid generation, will find it easier to grasp the implementation of these techniques when applied to moving mesh problems. The chapter concludes by introducing recent work by Budd et al [16],[18] and [17] which suggests that, where applicable, the moving mesh method should be chosen in such a way to mirror any self-similar properties lying in the underlying PDE. We will introduce some self-similar theory with specific reference to the porous medium equation (PME), for which an analytical solution exists in [64]. We shall take advantage of both these particular solutions and the self-similar

qualities in later chapters to verify our numerical computations.

We continue in Chapter 3 by describing the development of an efficient static moving mesh method from an existing technique known as *Contour Zoning*. The major advantage of the method is a notable reduction in computational effort via the use of one equation for a single value defined over a contour of the solution comprising several gridpoints. In two dimensions we adapt the contours in a static manner by moving them into positions governed by an easily enforceable equidistribution rule. We introduce the Contour Zoning method in one and two dimensions and in Chapter 4 use the method to compute numerical solutions to the PME and for a problem arising in semiconductor manufacturing. Our findings lead us to develop another moving mesh technique, initially in one dimension, motivated by the principle of conservation of mass.

Chapter 5 follows the development of this moving mesh technique. Here the underlying PDE is coupled with a local mass conservation law and the resulting single equation generates the speeds of the nodes from which the grids are found. The approximate solution to the PDE is then reconstructed from the current state of the mesh and a parameter $\theta(t)$ associated with the mass conservation via an algebraic relation. This method is then extended to couple the PDE with various styles of equidistribution, again with particular reference to the PME. This style of moving mesh implementation differs from the techniques described in Chapter 2 in the way that the coupled system is solved for in terms of the *grid speeds* alone and not in a standard interleaving approach for both the mesh and the solution. This approach is then extended further in Chapter 6 and numerical solutions are generated for the semiconductor problem and for a problem involving solution blow-up.

The penultimate chapter attempts to translate the success of the method in one dimension from the previous chapters to higher dimensions, with specific regard to the PME. We concentrate on the idea of mesh movement via mass conservation, employed to compute radially symmetric solutions. Several attempts are made to construct a genuinely two-dimensional algorithm using the same philosophy as in one-dimension. However, the results are subject to a greater or lesser extent to lateral oscillations in the solution, which we were unable to eliminate.

The final chapter summarises our findings and outlines possible areas of future research.

Chapter 2

Grid Generation and Moving Mesh Methods

In general, moving mesh methods are derived from introducing node speeds, i.e. velocities of computational nodes, into existing algorithms for generating computationally advantageous meshes for steady state problems. An obvious example of this development is the several variations of moving mesh partial differential equations (MMPDE'S) presented by Huang, Ren and Russell [44]. Here a simple equidistribution relation in one spatial dimensional is differentiated with respect to time in order to derive equations prescribing the correct velocities of nodes in order to preserve the equidistribution principle as the solution and grid evolve. In higher dimensions, due to the lack of a strict extension of the equidistribution idea in more than one spatial dimension, a popular idea is to evolve mesh speeds by attempting to keep a functional concerned with static grid generation minimal [47], [48].

In the first section of this chapter we shall explore some of the existing ideas in static grid generation so as to give a good understanding of the aims and methods behind many moving mesh methods. We then continue, by following how many of these methods are used to derive moving mesh techniques in one and two dimensions. The last section in the chapter details recent work in which moving mesh methods are chosen in accordance with theoretical properties of the underlying PDE.

2.1 Grid Generation Methods and Techniques

The simplest place to start an exposition of the basic philosophy behind the use of an adapted, irregular grid is in one dimension. The most widely used method is the equidistributed mesh, first introduced by De Boor [23] for obtaining good discrete approximations to continuous functions. The principles of the method were later applied to generating efficient computational grids for the numerical solution of steady PDEs. For example White [74] used a transformation to arc-length coordinates to generate equidistributing meshes for the numerical solution of two-point boundary value problems. Another approach is given in Denny & Landis [26], where the one-dimensional mesh was iterated by trying to reduce the truncation error of the solution of the underlying PDE after each iteration. This is a convenient point at which to formally introduce and define the equidistribution principle.

The main strategy behind the equidistribution idea is quite self-explanatory. The idea is to choose a mesh such that a measure of either the geometry of the represented function, or of the error of the numerical solution, is distributed equally between adjacent nodes. This measure is prescribed via a user-defined function known as the *monitor*, a positive-definite function of the solution u and/or its derivatives u_x,u_{xx}, of the form,

$$M = M(x, u, u_x, u_{xx}). \tag{2.1}$$

Later on in this section, we shall introduce various choices of monitor function and illustrate their effect on the resulting mesh. However, we begin by stating how this measure is distributed over the grid in a formal definition.

Given a mesh representing a physical space in one-dimension $x \in [a, b]$ with $N+1$ mesh points $x_i, i = 0, \ldots, N$, such that $x_0 = a$ and $x_N = b$, the equidistribution principle can be written

$$\int_{x_i}^{x_{i+1}} M dx = \frac{1}{N} \int_a^b M dx \quad i = 0, \ldots, N-1. \tag{2.2}$$

However, in most grid generation applications it is often more convenient to think of the equidistribution idea as one of a co-ordinate mapping from a computational

space to a physical one. The goal of the grid generation problem then becomes one of finding a suitable co-ordinate mapping or transform. This approach is common and forms the basis of most grid generation techniques and, indeed, moving mesh methods. Good explanations of this co-ordinate transform idea can be found in Baines [3] and Huang, Ren & Russell [44]. Concentrating still on one dimension, we define the computational space $\xi \in [0,1]$, so that the mesh points in physical space are related to the (usually regularly spaced) grid points ξ_i in the computational domain. Written formally, x is then a mapping from ξ to x

$$x = x(\xi).$$

Within this framework the equidistribution idea is written as

$$\int_0^{x(\xi_i)} M d\tilde{x} = \xi_i \int_0^1 M d\tilde{x} \tag{2.3}$$

or

$$\int_{x(\xi_i)}^{x(\xi_{i+1})} M d\tilde{x} = \frac{1}{N} \int_0^1 M d\tilde{x}. \tag{2.4}$$

Differentiating (2.3) with respect to ξ once gives the equation used by White [74], differentiating yet again yields the equation presented in Baines [3]

$$\frac{\partial}{\partial \xi}(M \frac{\partial x}{\partial \xi}) = 0. \tag{2.5}$$

Following this approach, the solution of (2.5) with Dirichlet boundary conditions

$$x(0) = a \qquad\qquad x(1) = b$$

produces an equidistributed grid for the given monitor function. However, equation (2.5) is non-linear since M depends not only on x but also on the solution u. To overcome this, an iterative approach is suggested using the algorithm

$$(M(x^p) x_\xi^{p+1})_\xi = 0 \qquad\qquad (p = 0, 1, ...)$$

which may be discretised in a semi-implicit style as follows,

$$M(x^p_{i+\frac{1}{2}})(x^{p+1}_{i+1} - x^{p+1}_i) - M(x^p_{i-\frac{1}{2}})(x^{p+1}_i - x^{p+1}_{i-1}) = 0. \tag{2.6}$$

The resulting tridiagonal system is easily solved using, for example, a Jacobi-iteration method.

When generating an equidistributed grid for good representation of a function or initial condition, the values of the monitor are known exactly and the iteration is usually quick and successful. However when using this type of iterative process for adapting a mesh to give a better numerical solution to an underlying differential equation, it is common to use an interleaving approach where the grid and solution are alternately updated, with the solution being interpolated between changing states of the mesh.

We now consider a few examples of possible monitor functions. The simplest such monitor, $M = 1$, produces an uniform equi-spaced grid. This monitor has been used in a moving mesh method with a moving boundary by Budd et al [16], as it permits attractive theoretical properties of the solution within the mesh movement, (details of which shall be discussed later in Section 2.3). Elsewhere, early work by Carey & Dinh [21] showed that minimising the error between a numerical approximation over a computational cell was equivalent to equidistributing the curvature monitor raised to a specific power, depending on which error norm was considered. However, the most common desired feature of using the technique of equidistribution is that the resulting grids have high mesh resolutions where solution gradients are steep and lower resolutions where the solution is less active. This in turn implies that the grid will then provide good approximations of derivative terms when using a suitable numerical scheme or solver. For this reason it is common for the monitor to involve derivative terms of the solution u. In this case, the simplest idea is to use the first derivative of u with respect to x, i.e.

$$M = |u_x|. \tag{2.7}$$

The effect of the gradient monitor on a monotonic function is that the solution values themselves become equi-spaced, since,

$$\int_{x_i} x_{i+1} M dx = u_{i+1} - u_i$$

see Figure 2.1. The most popular choice of monitor is the arc-length of the solution which has been used in many mesh generation and moving mesh methods (e.g. White [74], Dorfi & Drury [27]). The arc-length monitor is written as

$$M = \sqrt{1 + u_x^2}. \tag{2.8}$$

This monitor gives a smoother mesh overall than the gradient monitor, especially when encountering large variations in u, as shown in Figure 2.1.

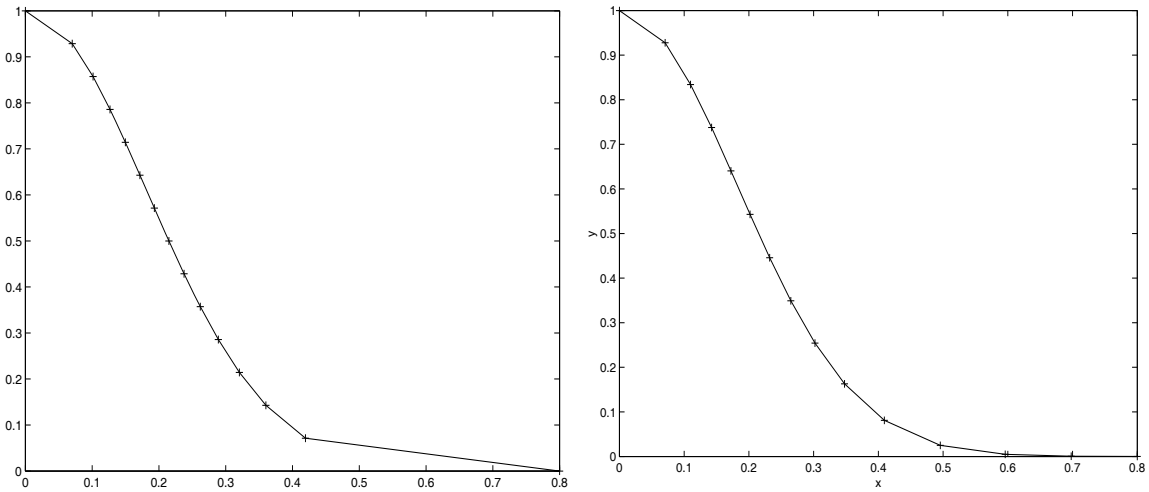

Figure 2.1: Examples of Grids using the gradient monitor (2.7) (left) and the arc-length monitor (2.8) (right).

In practice the derivative term in the arc-length monitor is often scaled by some parameter α, for example

$$M = \sqrt{1 + \alpha u_x^2}. \tag{2.9}$$

We shall comment more on the choice of monitor functions later on in Section 2.3.

Although equidistribution is the most common tool used when generating irregular computational meshes in one-dimension, the principle does not however extend strictly into two-dimensions and an alternative is needed.

One of the earliest, and most celebrated of such grid generation approaches in two dimensions is given in the appendix of Winslow's paper [77]. The main body

of which contains a method for the solution of a quasi-linear Poisson equation on a non-uniform triangular mesh, and the accompanying appendix outlines how to form such a mesh for regular domains. The ideas presented in this paper provide a basis for many of the higher-dimensional grid generation methods that followed. Once again, the approach is based on a mapping from a computational domain Ω_c to a physical domain Ω_p. The computational domain is represented as a regular equilateral triangular mesh composed of 2 sets of straight lines, associated with the inverse mappings $\xi(x, y)$ and $\eta(x, y)$ which satisfy the Laplace equations

$$\nabla^2 \xi = 0, \tag{2.10}$$

$$\nabla^2 \eta = 0. \tag{2.11}$$

The solution to (2.10 and 2.11), results in intersecting equi-potentials, i.e. $\xi = constant$ and $\eta = constant$, with the mesh completed using the intersections of the resulting sets of lines. The required mesh is found by inverting the transforms and putting them in terms of $x(\xi, \eta)$ and $y(\xi, \eta)$ using the Jacobian $J = x_\eta y_\xi - x_\xi y_\eta$, so that (2.10 and 2.11) become

$$\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} = 0, \tag{2.12}$$

$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} = 0 \tag{2.13}$$

where

$$\begin{aligned} \alpha &= (x_\eta^2 + y_\eta^2), \\ \beta &= (x_\xi x_\eta + y_\xi y_\eta), \\ \gamma &= (x_\xi^2 + y_\xi^2). \end{aligned}$$

These equations can be discretised by the finite difference method outlined in the main body of the Winslow article [77] and solved via an iterative successive over-relaxation algorithm. Due to the averaging property of the Laplace equation the

constructed mesh is in some sense smooth and is also easily applicable to quadrilateral meshes. Notice that the method is in no-way linked to a function or numerical solution represented on the grid. The purpose of this early grid generation algorithm is to produce grids adapted to a particular domain, the shape of which is imposed via boundary conditions used in conjunction with (2.12 and 2.13). Winslow's method as outlined above was adopted by Thompson et al [70] to generate meshes around multiple curvilinear bodies used in modelling flow over various shaped airfoils.

Brackbill and Saltzman [13] took advantage of the idea and extended the method by allowing discretionary control of various mesh properties such as the smoothness and the orthogonality of the grid. Their paper highlights that solving the Laplace equations (2.10 and 2.11) is equivalent to minimising the functional (2.14) below which relates to the smoothness of the mesh, over the computational domain Ω_c.

$$I_s = \int_{\Omega_c} [(\nabla \xi)^2 + (\nabla \eta)^2] dV. \tag{2.14}$$

Similarly, by solving the Euler equations associated with minimising the functional related to the orthogonality of the mesh (see (2.15)), an orthogonal grid is produced.

$$I_o = \int_{\Omega_c} (\nabla \xi . \nabla \eta)^2 dV. \tag{2.15}$$

In practice, Brackbill and Saltzman suggested the use of linear combinations of such functionals, with the preferences of the user implemented through choices of coefficients. The over-riding theme seems to be that as such properties can be measured they can also be controlled. In their paper the variational approach was used in conjunction with a numerical solution to a steady PDE and results show that, as the chosen functional is minimised, so too is the numerical error. Hence we see the development of the idea of a choice of functional in higher dimensions mirroring the effect of a monitor function in one-dimensional equidistribution.

The methodology of Winslow [77] and Brackbill & Saltzman [13] can be thought of as special cases of a more general framework outlined later by Huang & Russell [46], [47] and [48]. Specifically, [48] presents the following functional (2.16) as a general form of a grid adaptation functional,

$$I[\xi,\eta] = \frac{1}{2}\int_{\Omega_p} [\nabla\xi^T G_1^{-1}\nabla\xi + \nabla\eta^T G_2^{-1}\nabla\eta]dxdy \tag{2.16}$$

where G_1 and G_2 are given symmetric positive definite matrices, referred to as the monitor functions. The desired mesh transformation is derived from the solution to the associated Euler-Lagrange equations,

$$\nabla.(G_1^{-1}\nabla\xi) = 0, \tag{2.17}$$

$$\nabla.(G_2^{-1}\nabla\eta) = 0. \tag{2.18}$$

It is easy to see that by choosing $G_1 = G_2 = I$ this general methodology reduces to Winslow's original ideas. Moreover Huang & Russell [48] give forms of G_1 and G_2 which correspond to Brackbill's mesh generation method [12].

Equations (2.17) and (2.18), together with Dirichlet boundary conditions form a harmonic map from the physical to computational domains and the reliability of the method stems from the guaranteed existence and uniqueness of the transform, provided that the boundary of Ω_p is convex and that $G_1 = G_2$. Details can be found in Dvinsky [30]. Again, in [46], details of the specific monitor used in [30] are given, involving distances from a given surface. As an illustrative example, below is the 'arc-length-like' monitor presented in [48]

$$G_1 = G_2 = \frac{1}{\sqrt{1+\|\nabla u\|^2}}(I + \nabla u \nabla u^T). \tag{2.19}$$

Further work by Cao et al [20] proved by the use of Green's functions that the mesh can be aligned in certain directions and mesh concentrations can also be influenced in certain directions by controlling the eigenvectors and eigenvalues of the monitor matrices (specifically when $G_1 = G_2$). In particular, findings from this paper suggest that minimising the function I concentrates nodes in regions where the eigenvectors of G_1, λ_1 and λ_2 change significantly. This seems to have stemmed from earlier work by Brackbill [12] and Knupp [54]. The latter followed his own earlier work, this time combining the Winslow functional and another functional giving a certain amount of directional control over the grid by attempting to align mesh lines

with a prescribed vector field related to the approximate solution. Knupp [54] also used the variational approach to grid generation, using weights from sets of vector fields, with the resulting meshes aligning themselves with the same vector fields in some least-squares sense, of course some prior knowledge of the appropriate vector fields being needed.

Another interesting example of the application of Winslow and other such methods, is outlined in Farmer [32] for use in modelling geological features. Here grids are needed which honour 'control lines' representing features such as faults. These control lines are extended to the boundaries of the domain via interpolation, leaving the domain sectioned into several rectangular domains, which are then discretised using the outlined grid generation techniques.

This functional framework for finding the desired mesh transformation is a popular and convenient one, especially when used as the basis of a higher dimensional moving method, as we shall see later. For completeness, it is worth noticing that in one dimension, minimising the functional

$$I[\xi] = \frac{1}{2}\int_0^1 \frac{1}{M}\left(\frac{\partial \xi}{\partial x}\right)^2 dx$$

yields the equidistribution equation for a given monitor M [48]. Since this general framework has been developed to work as part of high dimensional moving methods, solution procedures for these methods incorporating the mesh movement process will be outlined later in Section 2.2.

Alternative two-dimensional analogues of equidistribution for grid generation can be found in Baines [3] and Huang & Sloan [50]. In the former paper an equation to solve for the appropriate monitor function is given as a natural generalisation of equation (2.5),

$$\nabla_\zeta (M(n)\nabla_\zeta n) = 0$$

where n is a coordinate along the direction of ∇u and $\zeta = (\xi, \eta)$. This translates into a 'local equidistribution in the direction of ∇u'. Replacing n by x or y gives the equations below, which are of the familiar Euler-Lagrange form presented earlier,

$$\nabla_\zeta(M\nabla_\zeta x) = 0,$$
$$\nabla_\zeta(M\nabla_\zeta y) = 0.$$

These equations are again solved with an interleaving approach with Dirichlet conditions. The resulting grid is unable to equidistribute M precisely but clusters grid points in regions of high M as desired. Further, Baines [4] shows that a least squares minimization of a residual of a vector field is equivalent to a least squares measure of equidistribution on triangular meshes, in some sense extending the work in one dimension by Carey & Dinh [21].

Elsewhere, the work of Huang and Sloan [50] follows ideas set out by Dwyer [31] and Catherall [22] and a local equidistribution is obtained by imposing the strict one-dimensional form over two sets of co-ordinate lines.

It is worth taking time to grasp an understanding of these grid generation techniques as a precursor to studying moving-mesh methods. As we shall see in the following section, many moving grid algorithms are based upon an underlying principle for constructing meshes with effective grid resolutions.

2.2 Moving Mesh Methods

In the previous section, we have outlined the aims and some techniques behind the generation of irregular grids. We now turn our attention to methods which aim to move the mesh in time to solve non-steady differential equations, whilst retaining the properties (and hence the numerical benefits) of the ideas presented above. We shall make constant reference to the techniques in Section 2.1, so it makes sense to follow the same order of events, starting with the use of the equidistribution principle in deriving moving mesh methods in one dimension.

An early incorporation of the equidistribution idea into a moving mesh method is outlined by Petzold [34]. Here a natural extension of the interleaving numerical solution approach for a stationary, adaptive grid is presented. Since the solution of the problem now develops with time, the equidistribution part of the

interleaving solution approach is undertaken at intervals, usually chosen by some pre-determined error measure, during the forward integration in time. In other words, at certain times throughout the numerical solution of the equation, the grid is re-equidistributed, hence moving the nodes throughout time, the solution on the new grid being found via some interpolation process. In a slight variation on this technique Blom et al [11] used a predictive step, re-equidistribute the grid using the prediction and then update the solution on the new grid. The update step is written in a Lagrangian form, involving the movement of the nodes in the redistribution, hence no interpolation step is required. The Blom approach bridges the gap between the static, regridding technique of Petzold, and more dynamic traditional moving mesh methods. The major difference between the two is the interpretation of mesh speeds included within the solution procedure. We continue this theme further and explore the various forms of this continuously moving mesh idea.

In contrast to the regridding idea, an early dynamic moving mesh technique was devised by Dorfi & Drury [27]. Here a separate equation for mesh speeds is developed via a function R to control mesh resolution which acts in the same way as a monitor function (despite no formal mention of equidistribution ideas). A simple relation between the speeds of the points $\dot{x}$ and R is solved in conjunction with the underlying PDE. Other early additional moving mesh equations include the work by Adjerid & Flaherty [1], who used a moving mesh equation within a finite-element framework to equidistribute the local discretisation error within the scheme. Petzold [65] followed the regridding approach with a more dynamic moving mesh method, the idea here being that using transformed pseudo-Lagrangian moving mesh co-ordinates, mesh speeds can be chosen so as to minimise the movement of the mesh in the transformed variables, so the solution in these co-ordinates is changing as slowly as possible for an easier numerical solution.

White [75] followed earlier grid generation work in one-dimension by using a moving mesh method based upon the transformation to arc-length type co-ordinates. Applications of early moving mesh methods include the work by Larrouturou [56], working on a flame propagation problem, a single mesh speed being derived for the entire grid, this velocity chosen to preserve thermal energy in the solution, the entire

grid is then moved as a rigid body. For the reader's interest, a review of some of the earlier moving mesh methods in one-dimension can be found in Hawken et al [40].

We now turn our attention to the work of Huang, Ren and Russell ([44], [43] and [68]). In contrast to the work by Dorfi & Drury the moving mesh equation is derived directly from the equidistribution principle. In [44] several moving mesh partial differential equations (MMPDE's) are derived in this manner, with the aims of the resulting algorithm being simple, easy to program and relatively insensitive to the choice of user-defined parameters. In all seven of these MMPDE's are constructed using three different approaches, the first two of which are motivated by equidistribution. Using the one-dimensional computational and physical co-ordinate systems as described in Section 2.1, two quasi-static equidistribution principles (QSEP's), are obtained by differentiating the integral form of the equidistribution principle (2.3) with respect to ξ once and twice respectively,

$$M(x(\xi,t),t)\frac{\partial}{\partial\xi}x(\xi,t) = \int_0^1 M(\tilde{x},t)d\tilde{x} \tag{2.20}$$

and

$$\frac{\partial}{\partial\xi}\left\{M(x(\xi,t),t)\frac{\partial}{\partial\xi}x(\xi,t)\right\} = 0. \tag{2.21}$$

To introduce node movement into the picture, time differentiation is undertaken. Several mesh movement equations have been produced by, for example Anderson [2], Hindman & Spencer [41] and Ren & Russell [68], the former two papers being early attempts with the transformation between physical and computational space, first in one, [41], and later in two dimensions [2]. However some of these earlier forms include time differentiation of the integral quantity

$$\theta(t) = \int_0^1 M(\tilde{x},t)d\tilde{x}.$$

Huang, Ren & Russell state, without supporting argument, that the quantity $\theta(t)$ or its time derivatives are too complicated to include in actual computation. However, by first differentiating the original equidistribution principle with time and then with ξ twice we obtain

$$\frac{d}{dt}\left\{\frac{\partial}{\partial\xi}\left(M\frac{\partial x}{\partial\xi}\right)\right\}=0$$

which can be written as (MMPDE1)

$$\frac{\partial}{\partial\xi}\left(M\frac{\partial\dot{x}}{\partial\xi}\right)+\frac{\partial}{\partial\xi}\left(\frac{\partial M}{\partial\xi}\dot{x}\right)=-\frac{\partial}{\partial\xi}\left(\frac{\partial M}{\partial t}\frac{\partial x}{\partial\xi}\right) \tag{2.22}$$

so giving a moving mesh equation without reference to $\theta(t)$. In the same paper an alternative set of moving mesh equations, MMPDE's 2-4, are derived by considering (2.21) and requiring that the mesh satisfy the condition at the later time $t+\tau$ (where $0<\tau\ll 1$) instead of at time t, i.e.

$$\frac{\partial}{\partial\xi}\left\{M(x(\xi,t+\tau),t+\tau)\frac{\partial}{\partial\xi}x(\xi,t+\tau)\right\}=0.$$

This equation is thought to be a strong enough condition to regularize the mesh movement by Huang et al. Substituting the expansions

$$\begin{aligned}\frac{\partial}{\partial\xi}x(\xi,t+\tau) &= \frac{\partial}{\partial\xi}x(\xi,t)+\tau\frac{\partial}{\partial\xi}\dot{x}(\xi,t)+O(\tau^2)\\ u(x(\xi,t+\tau),t+\tau) &= u(x(\xi,t),t)+\tau\dot{x}\frac{\partial}{\partial x}u(x(\xi,t),t)\\ &\quad +\tau\frac{\partial}{\partial t}u(x(\xi,t),t)+O(\tau^2)\end{aligned}$$

into (2.2) and dropping higher order terms gives MMPDE 2 (2.23), which in fact is MMPDE1 with an additional 'correction' term

$$\frac{\partial}{\partial\xi}\left(M\frac{\partial\dot{x}}{\partial\xi}\right)+\frac{\partial}{\partial\xi}\left(\frac{\partial M}{\partial\xi}\dot{x}\right)=-\frac{\partial}{\partial\xi}\left(\frac{\partial M}{\partial t}\frac{\partial x}{\partial\xi}\right)-\frac{1}{\tau}\frac{\partial}{\partial\xi}\left(M\frac{\partial x}{\partial\xi}\right) \tag{2.23}$$

.

The extra term is a measure of how well the current grid is equidistributed and hence MMPDE 2 moves the grid towards an equidistributed state even when M is independent of t. For this reason, terms involving $\frac{\partial M}{\partial t}$ are less important for MMPDE 2 than MMPDE 1, and disregarding these terms leads to MMPDE's 3 and 4 respectively, i.e,

$$\frac{\partial^2}{\partial \xi^2}(M\dot{x}) = \frac{1}{\tau}\frac{\partial}{\partial \xi}\left(M\frac{\partial x}{\partial \xi}\right) \tag{2.24}$$

and

$$\frac{\partial}{\partial \xi}\left(M\frac{\partial \dot{x}}{\partial \xi}\right) = \frac{1}{\tau}\frac{\partial}{\partial \xi}\left(M\frac{\partial x}{\partial \xi}\right). \tag{2.25}$$

The remaining MMPDE's (5-7) are devised by considering attraction and repulsion pseudo-forces between nodes. Here the mesh movement is specifically motivated by taking the monitor to be some error measure, so nodes are attracted together when the error is larger than average and repelled when the measure is below average. The error is then expressed as an integral over each cell, W_i, usually taking the form

$$W_i = \int_{x_i}^{x_{i+1}} M(\tilde{x}, t)d\tilde{x}.$$

MMPDE's (5-7) stem from this relation and all involve the correction term mentioned above, which seems to be a key term as it can determine the time-scale for the mesh movement and hence can be adapted to suit the problem in hand. Moreover since the correction term can be derived from the equidistribution idea, its inclusion in the latter mesh equations suggests that the error is evenly distributed over the mesh and the equidistribution and attraction/repulsion ideas are therefore thought to be closely related. Huang, Ren & Russell also provide theoretical analysis suggesting that the MMPDE's cannot produce instances where nodes cross paths when the MMPDE is solved exactly, indicating stability of the resulting meshes. The stability analysis follows early work by Flaherty et al [33]. In particular it is noted that for MMPDE 1, the mesh would be stable if the measure

$$L(t) = \max_{0\leq\xi\leq 1}\frac{M(x(\xi,0),0)}{M(x(\xi,t,0)}$$

were to remain bounded. However for most choices of M, $L(t)$ is likely to increase. Li et al [58] went on to discuss the stability of such moving mesh systems in greater detail.

The resulting equations (MMPDE's 1-7) have spawned a variety of work in various applications, sometimes with a common modification, that being the spatial smoothing of the monitor function M. Dorfi & Drury [27] and Furzeland et al [34] came to the conclusions in their early moving mesh work that when using finite-difference schemes to approximate derivative terms, in order to obtain 'reasonable' accuracy the mesh should be, in some sense, smoothed. Verwer et al [72] proved that smoothing the mesh is equivalent to smoothing the monitor function over the grid. Motivated by this work, Huang, Ren & Russell [43] use MMPDE's (3-7) with a smoothed monitor function $\tilde{M}$ defined at each node by

$$\tilde{M}_i = \sqrt{\frac{\sum_{k=i-p}^{i+p} (M_k)^2 \left(\frac{\gamma}{1+\gamma}\right)^{|k-i|}}{\sum_{k=i-p}^{i+p} \left(\frac{\gamma}{1+\gamma}\right)^{|k-i|}}}$$

where γ is a smoothing parameter and p is a non-negative integer referred to as the smoothing index which determines the range of the smoothing. These ideas provide a valuable tool in higher dimensions, since using a locally smoothed monitor function is considerably easier than smoothing the entire mesh separately. Moreover it is noted in [44] and [43] that MMPDE's 3 and 4 permit a possible extension to multi-dimensions.

Mackenzie [59] and Stockie et al [69] have both applied the smoothed moving mesh equations to PDEs in one-dimension and later to systems of hyperbolic conservation laws, where monitors were not only smoothed but combined to provide a moving grid on which to simulate the development of several time dependent variables. Mackenzie & Robertson [60] also used a mesh equation based upon equidistribution applied to a problem involving a phase change. Here a monitor based upon the asymptotic behaviour of the problem was used, clustering nodes around the moving interface, whilst the inclusion of a constant term also allowed sufficient nodes to be placed away from the region. Further applications of the MMPDE's (1-7) include work by Qiu & Sloan [67], who applied MMPDE 6 with the outlined technique of smoothing the monitor to Fisher's Equation. Interestingly enough, a new monitor was derived for specific use with reaction-diffusion problems (2.26) after arc-length and curvature monitors proved to be unsuccessful: this was

$$M(x,t) = \left[1 + \alpha^2(1-u)^2 + \beta^2(a-u)^2\left(\frac{\partial^2 u}{\partial x^2}\right)^2\right]^{\frac{1}{2}} \tag{2.26}$$

where α,β and a are user defined parameters.

Huang and Russell [45], also investigated the addition of artificial diffusion terms to the monitor as a means of smoothing, the resulting method satisfying a mesh crossing condition and allowing for possible extension to higher spatial dimensions.

A so-called Moving Mesh Differential-Algebraic Equation (MMDAE) is developed by Mulholland, Qiu & Sloan [63]. Instead of using the an MMPDE, the mesh movement is prescribed by a QSEP (2.20 and 2.21) written in terms of an algebraic equation involving the stationary grid points and the monitor function M. In fact the algebraic relation is the equidistribution relation written previously in Section 2.1, equation (2.6). This is coupled with the moving grid Lagrangian form of the underlying PDE and integrated forward in time using a first-order backward Euler method (used since these systems tend to be stiff). In [63], this technique is used in conjunction with a pseudo-spectral processing of the solution of hyperbolic problems. Qiu & Sloan [67] continue the work, comparing the method and in particular the stability with the established MMPDE 5 of Huang et al [44]. Of particular interest is the stability of the discrete solution of the steady-state solution to Burgers' equation by examining possible steady solutions arising from the two adaptive discretisations of the unsteady problem.

We now move on to moving mesh methods in higher dimensions. In the previous section we outlined a class of stationary grid adaption methods based upon minimising a mesh generation functional. As with the moving-mesh techniques in one dimension, we introduce mesh speeds into such a grid adaption method so as to preserve the properties of the grid as it moves in time. A popular way to introduce mesh speeds into the mesh functional approach is by use of the so-called gradient flow equations. Following the approach of Huang & Russell [47], a functional $I[\xi,\eta]$ is minimised over the computational domain Ω_c. One way to minimise I is to follow the steepest descent direction given by the first derivative of I. The following 'gradient flow' equations define a flow which converge to the equilibrium state at $t \to \infty$,

$$\begin{aligned}\frac{\partial \xi}{\partial t} &= -\frac{\partial I}{\partial \xi}, \\ \frac{\partial \eta}{\partial t} &= -\frac{\partial I}{\partial \eta}.\end{aligned}$$

In practice a modified version of these equations is used in [47], with the inclusion of the familiar correction term τ and the introduction of P, an operator on the underlying function space.

$$\begin{aligned}\frac{\partial \xi}{\partial t} &= -\frac{P}{\tau}\frac{\partial I}{\partial \xi}, \\ \frac{\partial \eta}{\partial t} &= -\frac{P}{\tau}\frac{\partial I}{\partial \eta}.\end{aligned}$$

The extra term P is used to choose more suitable directions than that of steepest descent with the τ terms allowing the user to choose a suitable time scale for the problem. It has already been noted in Section 2.1 that the functional approach in one-dimension can be shown to be equivalent to the equidistribution principle. Moreover the approach here can be shown to be similar to using MMPDE 5 [44], being based on the attracting and repellent forces of the monitor function. Indeed Beckett et al [6] used a similar version of the monitor outlined previously (2.26) in conjunction with a one-dimensional analogue of (2.29) for the solution of Burgers' equation. More recently MMPDE 5 has been used in two dimensions as part of an adaptive finite element method by Cao et al [19] for the solution of a combustion problem consisting of coupled non-linear reaction-diffusion equations.

Huang & Russell give multi-dimensional generalisations using this methodology for MMPDE's 4 and 6. Using this approach and the general grid generation functional (2.16), a suitable P is given in terms of the determinants of the two monitor matrices, i.e. $\tilde{g_1} = det(G_1)$ and $\tilde{g_2} = det(G_2)$, giving the resulting MMPDE

$$\frac{\partial \xi}{\partial t} = -\frac{1}{\tau\sqrt{\tilde{g_1}}}\frac{\partial I}{\partial \xi}, \qquad \frac{\partial \eta}{\partial t} = -\frac{1}{\tau\sqrt{\tilde{g_2}}}\frac{\partial I}{\partial \eta} \tag{2.27}$$

or

$$\frac{\partial \xi}{\partial t} = -\frac{1}{\tau\sqrt{\tilde{g_1}}}\nabla.(G_1^{-1}\nabla\xi), \qquad \frac{\partial \eta}{\partial t} = -\frac{1}{\tau\sqrt{\tilde{g_2}}}\nabla.(G_1^{-1}\nabla\eta). \tag{2.28}$$

As with solving for a stationary mesh, the actual computations are carried out after interchanging dependent and independent variables, giving

$$\begin{aligned}\frac{\partial \overline{x}}{\partial t} &= -\frac{\overline{x}_\xi}{\tau\sqrt{\tilde{g}_1}J}\left\{+\frac{\partial}{\partial \xi}\left[\frac{1}{Jg_1}(\overline{x}_\eta^T G_1 \overline{x}_\eta)\right] - \frac{\partial}{\partial \eta}\left[\frac{1}{Jg_1}(\overline{x}_\xi^T G_1 \overline{x}_\eta)\right]\right\} \\ &\quad -\frac{\overline{x}_\eta}{\tau\sqrt{\tilde{g}_2}J}\left\{-\frac{\partial}{\partial \xi}\left[\frac{1}{Jg_2}(\overline{x}_\eta^T G_2 \overline{x}_\eta)\right] + \frac{\partial}{\partial \eta}\left[\frac{1}{Jg_2}(\overline{x}_\xi^T G_2 \overline{x}_\xi)\right]\right\} \qquad (2.29)\end{aligned}$$

where J is the Jacobian of the co-ordinate transform.

Given this general framework, equivalent MMPDE's can be constructed using the various specific functionals described in Section 2.1. Dirichlet boundary conditions are preferred for the solution of (2.29) as this yields a unique solution, but for many problems this is not applicable since the boundary may not be stationary. Indeed, in some cases it is useful to moves nodes around the fixed boundary, for which many techniques are under investigation, the most popular being preserving a one-dimensional arc-length equidistribution of nodes on the boundary (see Huang & Russell [39], [49],Beckett et al [7]).

Huang and Russell [47] outline a familiar interleaving approach for the solution of the higher dimensional MMPDE combined with the underlying physical PDE as follows.

- Calculate the monitor functions G_1 and G_2 on the current mesh.
- Update the mesh at time $t + \Delta t$ by integrating the MMMPDE(2.29), keeping G_1 and G_2 constant.
- Integrate the physical PDE to get the solution at time $t + \Delta t$ using the mesh

$$\underline{x}(t) = \underline{x}^n + \frac{(t - t_n)}{\Delta t_n}(\underline{x}^{n+1} - \underline{x}^n)$$

  and mesh speed

$$\dot{\underline{x}}(t) = \frac{(\underline{x}^{n+1} - \underline{x}^n)}{\Delta t}.$$

- Choose a value of Δt_{n+1} for the next time step from the physical PDE.

As with their work in one-dimension, Huang, Ren & Russell suggest that the time correction term τ is preset by the user or determined by the development of the solution. However the choice of this value in one-dimension is relatively insensitive and it is thought to be so in higher dimensions also. Central finite difference discretisations are used by Huang & Russell along with a simple rectangular uniform reference mesh for the computational space. Again, extending the work carried out in one-dimension, the monitor is smoothed locally.

On reflection, the functional framework for multi-dimensional moving mesh methods gathers together all of the work described, both in grid adaption and one-dimensional moving grid techniques, since the strict equidistribution ideas in one-dimension can be written in terms of a functional and the moving mesh methods in higher dimensions are derived from a functional approach to grid adaption.

As an interesting aside, work by Demirdžić & Perić, [25] and [24] considers moving mesh methods from a more practical aspect. The authors suggest that many of the moving mesh algorithms before them induce error by not satisfying exactly any relevant conservation laws. Work is continued in [24] and mesh movement equations are derived for the solution of the Navier Stokes equations from a general scalar quantity conservation law. The fact that relevant physical quantities are conserved almost 'by construction' in the method is considered to be of utmost importance and is the driving force behind the moving grid.

Whichever approach is undertaken, a good understanding of the numerical techniques alone may not be good enough for the solution of some problems. We shall continue in the next section by introducing recent work which combines moving mesh methods and self-similar solution techniques, which suggests reasonable choices of monitor functions for certain problems. In particular we shall consider application to the solution of the PME, which we now describe in detail.

2.3 Self Similar Solutions and Mesh Movement

Recently there has been great interest in the connection between moving mesh methods and the numerical approximation of solutions to self-similar problems. The main

focus of attention has been on the PME in one-dimension (2.30) and the question of using appropriate moving mesh methods to suit its properties. Most of this work has been carried out by Budd et al ([15], [16], [18]). In this section we shall outline the main ideas behind the self-similar solutions of the PME and review the resulting moving mesh methods specifically derived for its solution. We begin by formally introducing the PME and highlighting some of its properties.

In one dimension, the PME is a non-linear diffusion equation of the form

$$u_t = (u^m u_x)_x \tag{2.30}$$

where $m > 0$.

Equation (2.30) arises from the study of the diffusion of gas through a porous medium under the action of Darcy's law relating velocity to pressure gradient. Another application appears in the modelling of the swarming of various insect species from which under certain conditions an analytic solution is available, as presented in Murray [64]. There is now a fairly complete existence theory for the equation, given initial conditions $u(0,x)$, these solutions take the form of travelling waves forming on a region of growing compact support $[s_-(t), s_+(t)]$, with $u = 0$ for $x \geq s_+(t)$ and $x/les_-(t)$. The solution conserves two important quantities, namely mass and centre of mass. Given a solution u exists, we have that

$$I(t) = \int_{-\infty}^{\infty} u dx > 0. \tag{2.31}$$

Then

$$\frac{dI}{dt} = \int_{-\infty}^{\infty} u_t dx = \int_{-\infty}^{\infty} (u^m u_x)_x dx = 0.$$

Hence I, or mass is conserved. Similarly, if $\bar{x}$ is the scaled centre of mass

$$\bar{x} = \int_{-\infty}^{\infty} x u dx,$$

then

$$\frac{d\bar{x}}{dt} = \int_{-\infty}^{\infty} x u_t dx = \int_{-\infty}^{\infty} x(u^m u_x)_x dx = -\int_{-\infty}^{\infty} u u_x dx = -\frac{1}{2}\int_{-\infty}^{\infty} (u^2)_x dx = 0.$$

Since both mass and centre of mass are conserved. The speed of the moving boundaries can be derived from the conservation of mass property. For instance, in the case of $m = 1$, the interfaces propagate with finite speed given by

$$\frac{ds_{\pm}}{dt} = -u(s_{\pm}, t)_x.$$

Hence given symmetric initial conditions, the solution will remain symmetric throughout time.

Figure 2.2 illustrates the behaviour of this class of solutions, the dashed line representing the progress of the Murray Solution for $m = 2$. The value of m influences both the speed of the moving boundary and steepness of the solution inside the interface. The larger the value of m results in a steeper evolving front and a slower rate of displacement.

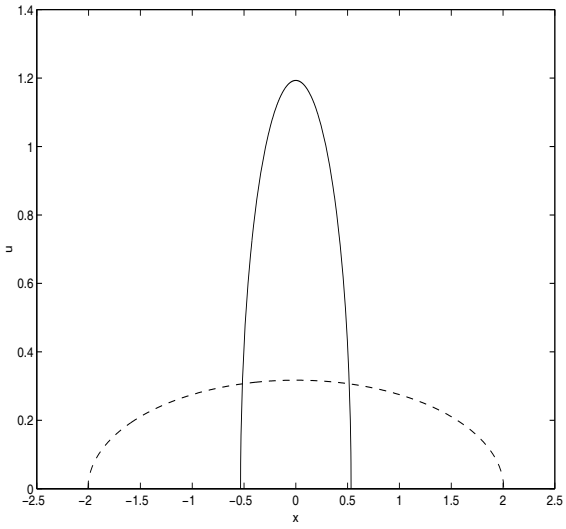

Figure 2.2: Evolution of solution to the PME (2.30), for $m = 2$.

Moreover the PME has a *scaling invariant* property which is the basis of the moving mesh ideas of Budd et al. We begin by providing a simple definition.

Given a system (u, x, t) satisfying a PDE, we introduce a mapping to a new system $(\hat{u}, \hat{x}, \hat{t})$ under the transformation

$$\hat{u} = \lambda^{\alpha} u, \quad \hat{x} = \lambda^{\beta} x, \quad \hat{t} = \lambda t \tag{2.32}$$

where λ is an arbitrary constant. The original system, (u, x, t), is said to be *scaling invariant* if the PDE under consideration is identical in both the original and transformed co-ordinates. Moreover, given a solution to the equation, if this too is invariant under the mappings then the solution is said to be *self-similar*. The benefits of such a transformation being available lies in the fact that sometimes the PDE is easier to solve in the transformed co-ordinate system. We continue with the derivation of such a transform for the PME.

Using the general form of the transforms (2.32), we can write the left hand side of the PME as

$$\begin{aligned} u_t &= \lambda^{-\alpha}\hat{u}_t \\ &= \lambda^{-\alpha}\hat{u}_{\hat{t}}\,(\lambda) \\ &= \lambda^{1-\alpha}\hat{u}_{\hat{t}}. \end{aligned} \tag{2.33}$$

Now considering the right hand side, substituting the transformations (2.32) gives

$$\begin{aligned} (u^m u_x)_x &= (\lambda^{-m\alpha}\hat{u}^m u_x)_x \\ &= \lambda^{-m\alpha}(\lambda^{-\alpha}\hat{u}^m\hat{u}_{\hat{x}}\hat{x}_x)_x \\ &= \lambda^{-\alpha(m+1)+\beta}(\hat{u}^m\hat{u}_{\hat{x}})_x \\ &= \lambda^{-\alpha(m+1)+\beta}\hat{x}_x(\hat{u}^m\hat{u}_{\hat{x}})_{\hat{x}} \\ &= \lambda^{-\alpha(m+1)+2\beta}(\hat{u}^m\hat{u}_{\hat{x}})_{\hat{x}}. \end{aligned} \tag{2.34}$$

Equating the two sides (2.34) and (2.35) gives

$$\lambda^{1-\alpha}\hat{u}_{\hat{t}} = \lambda^{-\alpha(m+1)+2\beta}(\hat{u}^m\hat{u}_{\hat{x}})_{\hat{x}},$$

and so, comparing the powers of λ, the system is scaling invariant provided that

$$1-\alpha = -\alpha(m+1)+2\beta$$

or

$$\alpha m - 2\beta + 1 = 0. \tag{2.35}$$

Furthermore we shall require the behaviour of the solution in the transformed coordinates to mirror the conservation properties of the solution in the original space. Taking into consideration equation (2.31),

$$I = \int_{-\infty}^{\infty} \lambda^{-\alpha} \hat{u} dx = \lambda^{-(\alpha+\beta)} \int_{-\infty}^{\infty} \hat{u} d\hat{x}.$$

So for the mass conservation property to be independent of λ, we require that

$$\alpha + \beta = 0,$$

which in conjunction with (2.35) gives us that the transformation (2.32) must satisfy

$$\alpha = \frac{-1}{m+2} \qquad and \qquad \beta = \frac{1}{m+2}. \tag{2.36}$$

Using the above conditions, the equation may be rewritten in terms of the transformed variables and solved to give a class of solutions to the PME. The resulting self-similar solution is invariant and hence independent of time, choosing $\hat{t}$ arbitrarily to be 1, we can write the transformed solution in terms of the original coordinates as

$$\hat{u} = u t^{\frac{1}{m+2}}, \quad \hat{x} = x t^{\frac{-1}{m+2}}. \tag{2.37}$$

The resulting, steady, differential equation can be solved by techniques found in Barenblatt [5] and Dresner[28], which form the basis of the solution given in Murray.

As mentioned above, the Murray solution arises from a model of insect dispersal. The PME comes from the model by assuming that the diffusion of the species population Q from a central origin is due to population pressure. The exact analytical solution is of the form

$$u(x,t) = \left\{ \begin{array}{ll} \frac{1}{\lambda(t)}[1 - \{\frac{x}{r_0\lambda(t)}\}^2]^{\frac{1}{m}} & |x| \leq r_0\lambda(t) \\ 0 & |x| > r_0\lambda(t) \end{array} \right\} \tag{2.38}$$

where

$$\lambda(t) = \left(\frac{t}{t_0}\right)^{\frac{1}{(2+m)}},$$

$$r_0 = \frac{Q\Gamma(\frac{1}{m} + \frac{3}{2})}{\pi^{\frac{1}{2}}\Gamma(\frac{1}{m} + 1)}$$

and

$$t_0 = \frac{r_0^2 m}{2(m+2)}$$

with $r_0\lambda(t)$ representing the position of the moving front, and $\Gamma(x)$ denoting the gamma function.

Upon consideration of the general higher-dimensional form of the PME,

$$u_t = \nabla.(u^m \nabla u),$$

Murray also provides a radially symmetric solution in two dimensions. Hence after transforming to radial coordinates, the PME becomes

$$u_t = \frac{1}{r}(r(u^m)u_r)_r \tag{2.39}$$

and we have the solution

$$u(x,t) = \begin{cases} \frac{1}{\lambda^2(t)}\left[1 - \{\frac{r}{r_0\lambda(t)}\}^2\right]^{\frac{1}{m}} & |r| \leq r_0\lambda(t) \\ 0 & |r| > r_0\lambda(t) \end{cases} \tag{2.40}$$

where

$$\lambda(t) = \left(\frac{t}{t_0}\right)^{\frac{1}{2(1+m)}},$$

$$r_0^2 = \frac{Q}{\pi}\left(1 + \frac{1}{m}\right)$$

and

$$t_0 = \frac{r_0^2 m}{4(m+1)}.$$

Similarly there exists a self-similar transformation for this radial case. Using the same forms for the transformation as used previously, we have

$$\hat{u} = \lambda^{\alpha} u, \quad \hat{r} = \lambda^{\beta} r, \quad \hat{t} = \lambda t. \tag{2.41}$$

Upon substitution into the radial form of the PME (2.39), when comparing powers of λ, the same relation found in the one-dimensional case is found (2.35). Upon considering the appropriate form for mass conservation, we have that if

$$I = 2\pi \int_{-\infty}^{\infty} r u dr$$

and

$$\begin{aligned} \frac{dI}{dt} &= 2\pi \int_{-\infty}^{\infty} r u_t dr \\ &= 2\pi \int_{-\infty}^{\infty} (r u^m u_r)_r dr \\ &= 0. \end{aligned}$$

So, upon substitution of the relevant transforms we have that

$$\begin{aligned} I &= 2\pi \int_{-\infty}^{\infty} \lambda^{-\beta} \hat{r} \lambda^{-alpha} \hat{u} dr \\ &= 2\pi \lambda^{-(\alpha+\beta)} \int_{-\infty}^{\infty} \hat{r} \hat{u} \lambda^{-\beta} d\hat{r} \\ &= 2\pi \lambda^{-(2\alpha+\beta)} \int_{-\infty}^{\infty} \hat{r} \hat{u} d\hat{r} \end{aligned}$$

which, upon comparison of powers of λ gives

$$\alpha + 2\beta = 0. \tag{2.42}$$

Equations (2.35) and (2.42) give us with the associated self-similar transformation in the radial case, provided that

$$\alpha = \frac{-1}{m+1}, \qquad \beta = \frac{1}{2m+2} \tag{2.43}$$

which, in turn, provides the invariant radial solution in terms of the original coordinates

$$\hat{u} = ut^{\frac{1}{m+1}}, \quad \hat{r} = rt^{\frac{-1}{2m+2}}. \tag{2.44}$$

Although we shall not be directly making use of these transformations in the numerical methods to be introduced in later chapters, the existence of the theory allows us to validate approximate solutions for the PME in both the radial and one-dimensional cases. Our aim is that the approximate solutions will, under the appropriate transform, display the invariant properties of the exact solutions.

Having demonstrated that the true solution to the PME has a self-similar solution under the prescribed transformation, Budd et al conclude that the numerical scheme must inherit the self-similar properties of the analytic solution, specifically a moving mesh should be used for which the monitor is scaling invariant too. In [16] Budd et al implement a moving mesh method involving the simplest of monitors which preserves the said properties, this being $M = 1$. Using finite difference approximations to the pseudo-Lagrangian form of the PME with $m = 1$ and the speeds of the moving boundary, the resultant mesh has an expanding uniform resolution. The method is applied to the PME in both the original and scaled variables and it is found that the scaling resulting from the discrete self-similar solution is identical to that in the continuous case. Hence the discrete self-similar solution has the dynamics of the underlying solution in the original variables. The scheme used permits the conservation of mass (and the centre of mass) and it is shown that the resulting discrete self-similar solution converges to the true self-similar solution as the number of nodes in the mesh is increased. Later Budd & Piggott [18] suggest that when using the moving mesh PDE to solve scale invariant problems, the monitor should also in some sense be scaling invariant. Another obvious choice is the mass monitor,

$$M = u.$$

which we shall use later in this thesis. Budd & Piggots's results show that, when $m = 1$ and the evolution of the PME is fairly gentle, the mass monitor preserves all the desired properties and is able to model the shallow front formation behaviour at the moving boundaries. No higher values of m are considered in [16], for which the front at the moving boundary is considerably steeper and a more suitable monitor

would be needed to resolve the front. Budd et al suggest that, since the monitor is chosen in such a way as to exhibit the properties of the true and self-similar solution, both the discrete solution and the moving mesh method also permit this behaviour.

Moving away from the PME, similar ideas have been applied to the non-linear Schrödinger equation [14] and to problems with blow-up [17]. In the latter, similar results are achieved, even when the smoothing process suggested by Huang, Ren & Russell [43] is applied to the chosen invariant monitor.

It is hoped that this chapter has provided a brief, yet informative, insight into existing techniques for grid adaption and in particular moving mesh methods. Although we shall not be implementing any of these methods directly, some of the ideas explored in this chapter will be of considerable relevance when we come to derive our own techniques as the thesis continues.

Chapter 3

Contour Zoning

In this chapter we outline the development of the Contour Zoning method [52] for the efficient approximate solution of PDEs. In two or higher dimensions, the most attractive feature of the method is that it reduces the solution of the underlying PDE to one dimension by grouping non-contiguous 'zones' of mesh points together and solving for a constant value over each zone. Moreover the method, in which computational nodes are collected and equidistributed together imposes a higher-dimensional equidistribution principle.

To begin with, we trace the development of the method from the initial use of a stationary rectangular mesh to that of a moving triangular grid, whilst maintaining the two attractive properties of the algorithm stated above. The resulting form of the method, when translated into one dimension, represents that of a regridding algorithm such as those described in Section 2.2. In Section 3.2 we introduce the method in one-dimension version in which the equidistribution is exact with respect to a gradient monitor. We also introduce an adaptive time step with a link to the scaling invariance of the PME following as discussed by Budd et al and outlined in Chapter 2.3. In Section 3.3 we explain in detail the full contour zoning solution algorithm and the generation and movement of the triangular grids involved in two-dimensional problems.

3.1 History and Development

The Contour Zoning method was first introduced into a prototype code developed at AEA Winfrith approximately fifteen years ago and trial results were published in nuclear engineering literature [37] & [38]. In another later reference written by Inston [52] the Contour Zoning method was outlined in two spatial dimensions using a fine scale regular rectangular computational mesh.

At the start of each time step, a coarse mesh solution was interpolated onto a fine-scale mesh. The range of the field variable u was then split up into equal intervals. Grid points with solution values lying in the same intervals were collected together into a zone. Each zone was then assumed to have a constant value for u and equations were derived for determining the constant value in each zone by summing flux terms out of each section of mesh relating to the boundary of the zone. In some sense these zones were grouped together to form a large control volume in finite volume terms. After the solution for the constant values, new zones were formed by updating the coarse mesh solution, and again equally dividing the interpolated solution range onto the fine mesh.

In subsequent work (Blake [8] and Thuburn [71]) it was noted that in its original form the contour zoning equations were non-convergent, due to poor approximations to the outward normal derivatives at the boundaries of the zones. More specifically, the spatial step size used to approximate derivatives was dependent on the resolution of the underlying fine-scale mesh and not on the size of the zones in question. These inaccuracies were corrected with the use of a more suitable step size and a connection made between the equal interval choice of zones and a weak form of equidistribution in higher dimension of a monitor function involving the first spatial derivative of u.

The method was then extended onto an unstructured contour-based triangular grid in [10]. This was originally intended to give the method more freedom, both in a more accurate representation of the zones and in any possible future attempts in moving or re-shaping of the zones. The latter reason became more important, especially after the departure of the coarse mesh solution which did, albeit clumsily, change the structure of the zones. In order to avoid the same poor approximations found in [52], it became clear that distances between similarly valued nodes would

have to be traversed by a single set of triangles, the result being that similarly valued nodes, i.e. the zones, now became the actual contours of the solution. Finally it followed that by working on the resultant triangular grid the representation of the solution would naturally become piecewise linear between the contours. The modified method was found to be first order accurate in space.

Mesh movement was introduced into the triangulated grid in [10]. The equidistribution principle mentioned above is used to shift nodes and hence contours to more advantageous positions within the grid after each solution update. Despite the equidistribution involving the simplest of monitor functions, the ease of implementing this idea complements the reduction in the size of the system to solve for the values of the contours involved. Technically the algorithm involves the location and interpolation over edges connecting adjacent contours, but further savings are achieved by retaining connectivity between grid points throughout. It is this final state of the method which we shall outline in detail in Section 3.3. However to start with we shall now explain a form of the algorithm in one dimension to give the reader a easier introduction to the method and to more easily highlight its prominent features.

As an aside, it is worth noting that a contour based algorithm had been developed previously (see Dritschel [29] for details) with the intended application being in numerical weather prediction. However it differed from the Contour Zoning approach in that contours were moved around a grid by constructing a velocity field with the intended application being in numerical weather prediction.

3.2 Regridding in One Dimension

We now outline the contour zoning method in one dimension. Obviously in one dimension there will be no reduction as such, in the number of equations to solve since the 'contours' are now simply individual nodes. However the simple implementation and exact preservation of the equidistribution idea, which powers the grid movement, is easily demonstrated in this simple case. As previously stated, in one dimension this work is strongly reminiscent of work by, for example, Petzold

[34] mentioned in Chapter 2. We shall begin by covering the finite volume solution of the underlying PDE and then move on to the use of the first derivative monitor function and the implementation of the grid movement. We shall finally describe a novel way of choosing an adaptive step size with reference to the time-scaling invariance ideas of Budd et al as covered in a later section of the previous chapter.

3.2.1 Solution of the PME

We now derive the finite volume scheme for an underlying PDE of non-linear diffusion form, with diffusion coefficient of the form $D = D(u)$ and we begin by restricting ourselves to only one spatial dimension. Hence we have

$$u_t = (D(u)u_x)_x. \tag{3.1}$$

Since we are using a finite volume construction on a mesh over a region $[a, b]$ consisting of $N+1$ nodes, $x_0, \ldots, x_N$, we first define the control volume associated with node x_i. In the standard fashion, this is chosen to be defined as the region captured between the midpoints of the two adjacent cells, as shown in Figure 3.1. We consider the integral form of (3.1) over this region,

$$\int_{x_i-\frac{1}{2}}^{x_{i+\frac{1}{2}}} u_t dx = \left[Du_x\right]_{x_i-\frac{1}{2}}^{x_{i+\frac{1}{2}}}. \tag{3.2}$$

Discretising (3.2) semi-implicitly, using linearly interpolated expressions to represent the diffusion coefficients at the midpoints of cell we have

$$(x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}) \left[\frac{u_i^{n+1} - u_i^n}{\Delta t}\right] = D^n_{i+\frac{1}{2}} \left[\frac{u_{i+1}^{n+1} - u_i^{n+1}}{x_{i+1} - x_i}\right] - D^n_{i-\frac{1}{2}} \left[\frac{u_i^{n+1} - u_{i-1}^{n+1}}{x_i - x_{i-1}}\right]$$

which, when rearranged, yields the system of equations,

$$-\frac{D^n_{i-\frac{1}{2}}}{x_i - x_{i-1}} u_{i-1}^{n+1} + \left[\frac{D^n_{i-\frac{1}{2}}}{x_i - x_{i-1}} + \frac{D^n_{i+\frac{1}{2}}}{x_{i+1} - x_i} + \kappa\right] u_i^{n+1} - \frac{D^n_{i+\frac{1}{2}}}{x_{i+1} - x_i} u_{i+1}^{n+1} = \kappa_i u_i^n \tag{3.3}$$

for $i = 0, \ldots, N$, to be solved at each time step, where

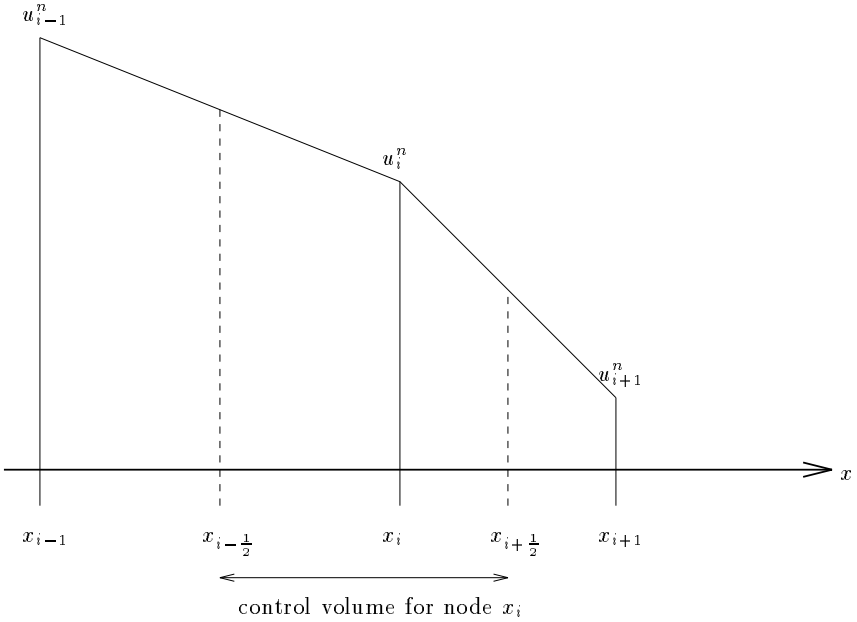

Figure 3.1: Construction of Finite Volume Approximation

$$\kappa_i = \frac{x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}}{\Delta t} = \frac{x_{i+1} - x_{i-1}}{2\Delta t}.$$

The system is completed by the appropriate discretisations of boundary conditions. Due to the semi-implicit discretisation of the diffusion coefficients the system is linear. Moreover, the system will also be tridiagonal and hence can be solved quickly with a suitable solver. A maximum principle which exists for this type of semi-implicit style discretisation guarantees the avoidance of new extrema as the solution steps forward in time [62].

For the solution of the PME, due to the rapid change in the diffusion coefficient with time, it is advantageous to use an adaptive time step and we shall the discuss the choice of the step size in Section 3.2.3.

3.2.2 Moving the Grid in One Dimension

We now illustrate how grid points are moved in one dimension by the contour zoning method. Initially, the gridpoints are still chosen by equally dividing the solution range. We first show that when the solution is monotonically decreasing, this is

equivalent to enforcing an equidistribution property exactly with the monitor function chosen to be

$$M(u) = u_x. \tag{3.4}$$

Starting with the equidistribution principle, as defined in Section 2.1, and working on a grid spanning the domain $x \in [a, b]$ consisting of $N + 1$ nodes $x_i, i = 0, \ldots, N$, we have that

$$\int_{x_i}^{x_{i+1}} u_x dx = \frac{1}{N} \int_a^b u_x dx \quad i = 0, \ldots, N-1$$

which simplifies to

$$u_i - u_{i+1} = \frac{1}{N} (u_0 - u_N) \quad i = 0, \ldots, N-1.$$

It follows that grid points chosen in this way will satisfy the equidistribution principle for the chosen monitor. The simplicity of the resulting grid makes this distribution easy to enforce and stems from the solution being monotonic. If it were not monotonic, then some iterative solution would be needed to avoid multiple grids points being defined for a single solution value.

Initially, given a suitable invertible function describing the initial state $u(x, 0)$, points on the u axis are found by simply calculating the values of u_i from

$$u(x_i, 0) = u_i = u(a, 0) - \frac{i}{N} (u(a, 0) - u(b, 0)) \quad i = 0, \ldots, N \tag{3.5}$$

where $u(a, 0) = u(x_0, 0)$ and $u(b, 0) = u(x_N, 0)$ are the maximum and minimum values respectively of $u(x, 0)$ over the domain. Then, to obtain the initial state, the grid points are found by inverting the function $u(x, 0)$, i.e.

$$x_i = u^{-1}(u_i, 0).$$

As the solution progresses, we can still use the relation (3.5) by using interpolation on the piecewise linear approximation to u to find the positions of the corresponding grid points. Figure 3.2 illustrates this procedure using only 3 grid points. Given a newly updated solution $\acute{u}^n$ at grid points x^n, we can use these new

values defined on the existing mesh to calculate, from (3.5), the solution value of the re-equidistributed grid point $\acute{u}_i^n$. Using linear interpolation from the appropriate section of the current solution, we find the corresponding new grid position $\acute{x}_i$.

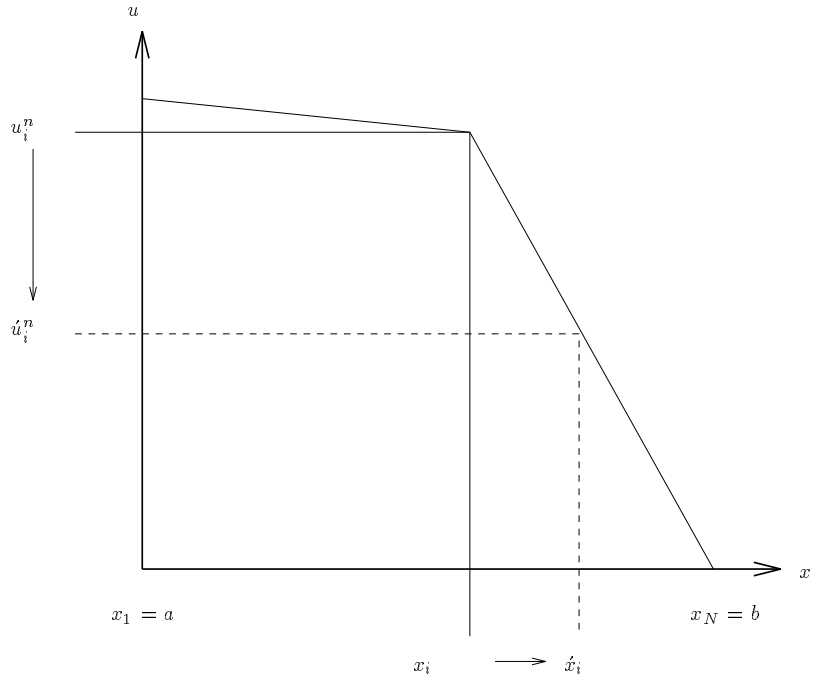

Figure 3.2: Moving nodes using linear interpolation

Figure 3.3 illustrates the steps above on a larger mesh to show the overall effect of the grid movement. Starting with a freshly equidistributed set of nodes and solution values in the top graph, the solution of the PDE is updated at the next time step in the middle figure. The new equidistributed solution values are computed from them, via interpolation, and from the updated solution the positions of the new nodes are found (shown in the bottom section). Figure 3.3 also demonstrates where, in the solution, the monitor function (3.4) positions nodes. It is obvious from the figure that grid points are clustered in regions only where the gradient is high, as expected.

In many existing regridding algorithms the time at which relocation of mesh points is carried out is prescribed by some error indicator, to save on computational cost by only regridding when it is needed. However, in keeping with the original contour zoning approach and since when using this monitor the grid movement process is simple, we can afford to reposition nodes after each update of the solution.

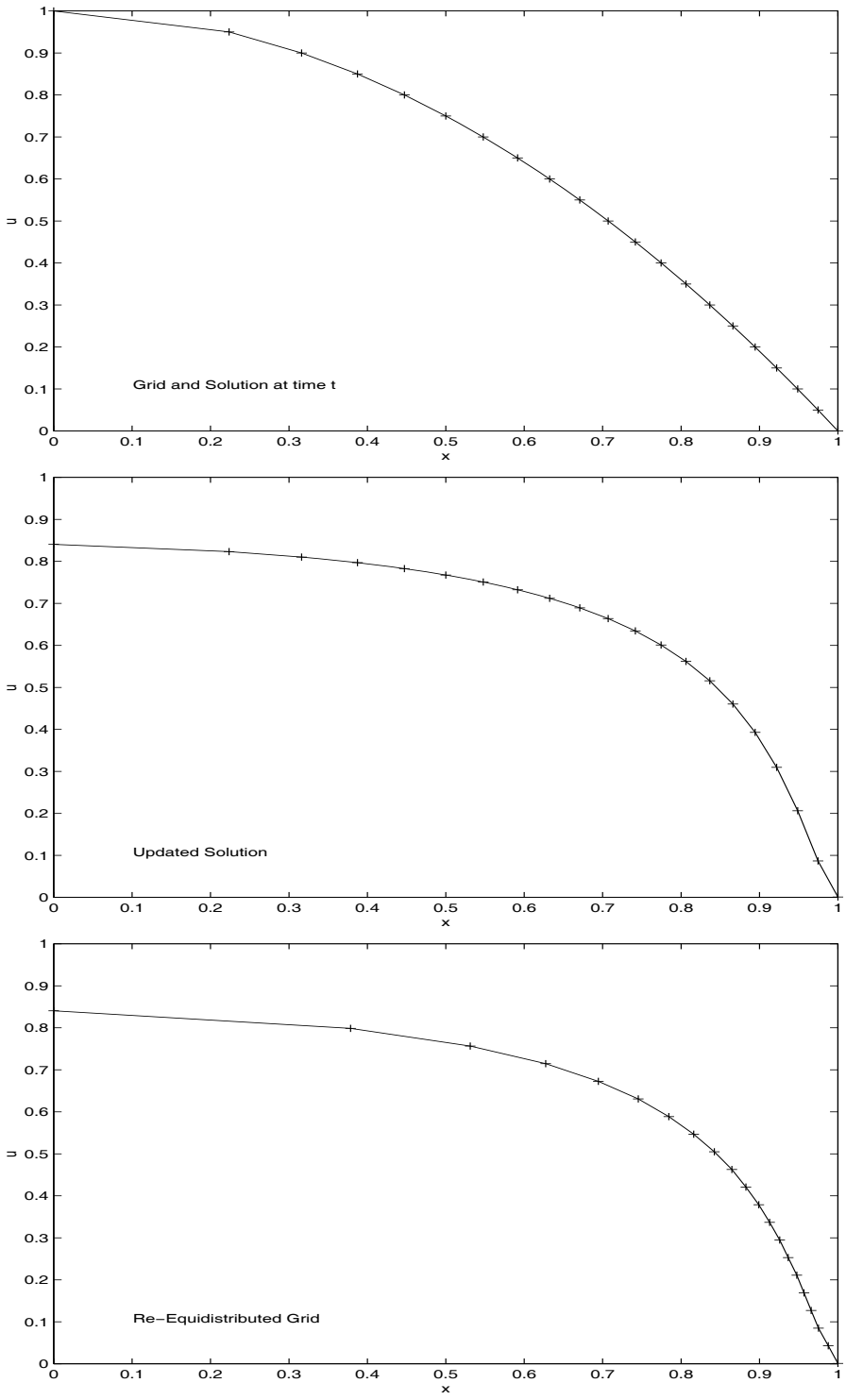

Figure 3.3: The three stages of node movement

We continue now by suggesting a choice for an adaptive time step.

3.2.3 Choosing a Time Step

The solution of the one-dimensional porous media problem develops with different speeds at different stages of its evolution. Due to the non-linearity of the diffusion co-efficient the rate at which the solution diffuses at the early stages of the solution is much quicker than at later times. By using a constant time step size Δt in the above algorithm, the finite volume solution may not capture this early behaviour very accurately and hence incur large truncation errors in the long-term solution. Hence it becomes obvious that we need to choose an adaptive time step in tune with the porous-media equations natural time scaling. Our choice of step length then, should be small when diffusion is fast and then the step length can be increased as activity dies down. For simplicity we choose to measure the rate of diffusion at the local maximum, which, as our solutions are monotonically decreasing will conveniently be at $x_0 = a$. It is possible to estimate the current rate of diffusion by using a local explicit solution for u at the maximum. In summary, we choose our step length so that the maximum value of u will decrease after each update by roughly the same amount, chosen by the user and denoted as δu. We can write the non-linear diffusion equation discretised explicitly as

$$\frac{|u_0^{n+1} - u_0^n|}{\Delta t} = \left| D(u_{\frac{1}{2}}^n) \frac{(u_1^n - u_0^n)}{(x_1 - x_0)^2} \right|.$$

Substituting $\delta u = |u_0^{n+1} - u_0^n|$, an averaged term for the diffusion coefficient, and re-arranging we have an expression for the step size, namely,

$$\Delta t = \left| \frac{\delta u (x_1 - x_0)^2}{D(u_{\frac{1}{2}}^n)(u_1^n - u_0^n)} \right|. \tag{3.6}$$

We can use the self-similar theory from the previous chapter to support this choice of time step in the case of the PME. Equation (2.37) gives the invariant, self-similar solution in the original coordinates. We can rewrite the original solution u as

$$u = \frac{\hat{u}}{t^{\frac{1}{m+2}}}$$

where $\hat{u}$ is invariant. Figure 3.4 illustrates the behaviour of the solution taken at an arbitrary point $(\hat{x}, \hat{u})$. It is easy to see that by taking a constant decrease in u over each time step, the resulting increments in time will be small where decay is rapid and will become relaxed as the behaviour dies down. For the radially symmetric problem the transformed solution has a similar form so the adaptive time step approach is also valid. In practice a maximum and/or minimum time step may be imposed.

To summarise, given a current solution at time t, the algorithm is implemented as follows :

- Choose the appropriate step size Δt using (3.6).
- Integrate the underlying PDE forward to time $t + \Delta t$ by solving the system of equations (3.3), with an appropriate choice of Δt.
- Using (3.5) find the values of the re-equidistributed grid points $\acute{u}^n$.
- Find the new grid points $\underline{\acute{x}}$, via linear interpolation, then return to the first step and repeat until the time integration of the PDE is complete.

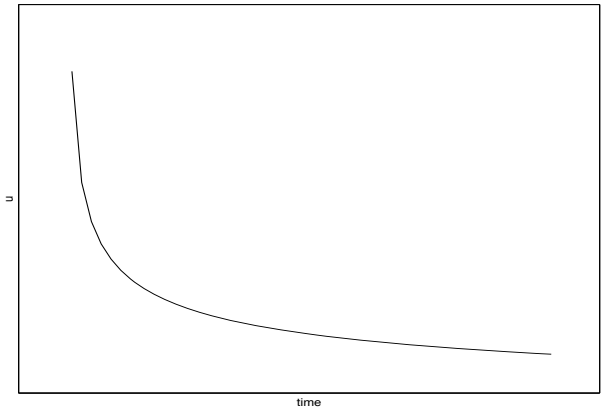

Figure 3.4: Decay of u with time at an arbitrary point

3.3 Contour Zoning and Moving Grids

We now move onto describe the full Contour Zoning method to be used in higher dimensions. As explained in Section 3.1, the method in two dimensions works on a contour based triangular computational grid. Many of the ideas could be extended to provide approximate solutions in even higher dimensions, but in this section we stay with working in two spatial dimensions.

The reduction of the number of equations to be solved in the original basic contour zoning idea now applies, having one equation to solve for the 'height' of each contour. Each contour consists of a number of nodes, hence the saving in effort is preserved. We shall continue this section by giving an insight into the grids used for the computations and their construction.

3.3.1 Contour Based Triangular Meshes

All of the two-dimensional calculations presented take place on unstructured triangular computational grids based on discretised contours of the function u. An initial grid is generated from the contours of the initial conditions of the problem, since the solution evolves from this state and no extra nodes are added or points taken from it. We illustrate the features of the grid with the construction of the initial mesh.

First of all, the heights of the initial contours are to be decided. In keeping with the original contour zoning ideology, this again is done by equally dividing the solution range, so formally this is similar to (3.5) in Section 3.2.2. Although this no longer corresponds to a strict equidistribution idea as in one-dimension, in choosing the contours in this way, the contours, and hence nodes, will also be clustered in regions of steep gradient.

Once the heights of the contours are chosen, the next step is to form the contours and then to discretise them with grid points. This is easily done using a MATLAB contour plotting routine that returns points lying on specified contours. Given a user specified number of points to lie on each contour and using linear interpolation along the co-ordinates given by MATLAB, nodes can be equally spaced along each contour.

Finally a triangulation is constructed between each pair of adjacent discretised contours. The type of triangulation used is thought to be of little significance, since the solution values of the connected nodes can only take one of two values of the heights of the contours in question, and hence the spatial derivatives between the contours are somewhat limited. The method of triangulation used here is by Delaunay (see [61]), although it may be necessary to remove any triangles connecting nodes belonging to the same contour. These are not required, since spatial gradients along contours always equal zero.

Figure 3.5 shows an example of such a grid. Here the grid is generated for a radially symmetric piecewise continuous initial condition typical of the family of solutions found in the insect dispersal problem [64]. In this relatively simple example, 21 contours are discretised (including the maximum at $(0,0)$), and each contour contains 15 nodes. The radial cross-section on the left hand side of the figure illustrates how the heights of the contours are equally spaced on the solution range.

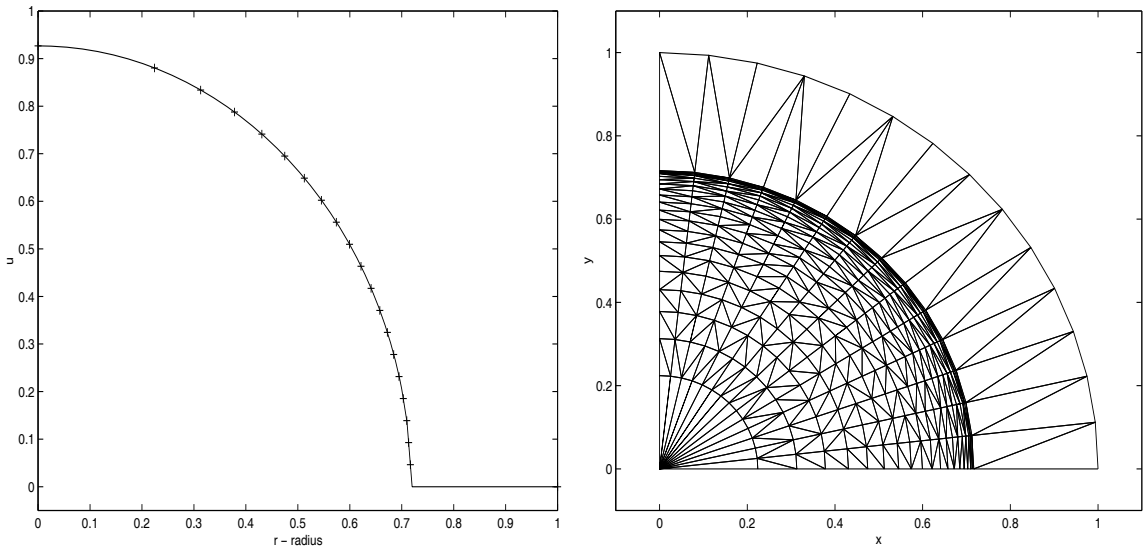

Figure 3.5: Example of Contour Based Grid with associated radial cross-section

Of course, there is no strict way to construct these grids. Various parameters can be altered, for instance equal numbers of nodes do not have to be placed on each contour. Instead an incremental step length along the contours could be chosen, and nodes spaced equally along each contour using this step length. This would

not alter the solution technique, as the connectivities are kept constant as the grid develops.

In the next section we derive the two-dimensional contour zoning equations used to update the values of the heights of the contours, as prescribed by the underlying PDE.

3.3.2 Contour Zoning Solution Technique

The formulation of the Contour Zoning Equations over a triangular mesh is largely an amalgamation of the ideas presented in the original Contour Zoning literature [52] and the solution techniques of Winslow [77]. As in the case of the one-dimensional algorithm, and so as to cover all problems attempted later, we consider the non-linear diffusion equation. In two dimensions we have

$$\frac{\partial u}{\partial t} = \nabla.(D\nabla u) \tag{3.7}$$

where $D = D(u)$ is the non-linear diffusion coefficient.

Now consider an interior node in the triangular mesh. We define a secondary mesh of irregular polygons whose vertices are alternately the centres and the mid-points of the sides of the adjacent triangles to which those of the node in question belongs. An example of this mesh structure is shown in Figure 3.6, the secondary mesh element for the node i being denoted by the broken lines. For computational ease it is useful to notice that when the definition of the secondary mesh is complete, each of the primary mesh triangles will be divided into three quadrilaterals of equal sizes.

Secondary elements of nodes that lie on the same contour are collected together to form the control volume of the contour. It is this area, V_{iz}, which is integrated around when considering the integral form of (3.7) for the iz^{th} contour, namely

$$V_{iz}\frac{\partial u}{\partial t} = \int\int_{V_{iz}} \nabla.(D\nabla u)dxdy.$$

By Gauss' theorem the right hand side of this equation is equal to the integral around the boundary of V_{iz} of the normal flux of the diffusing quantity u. Substitut-

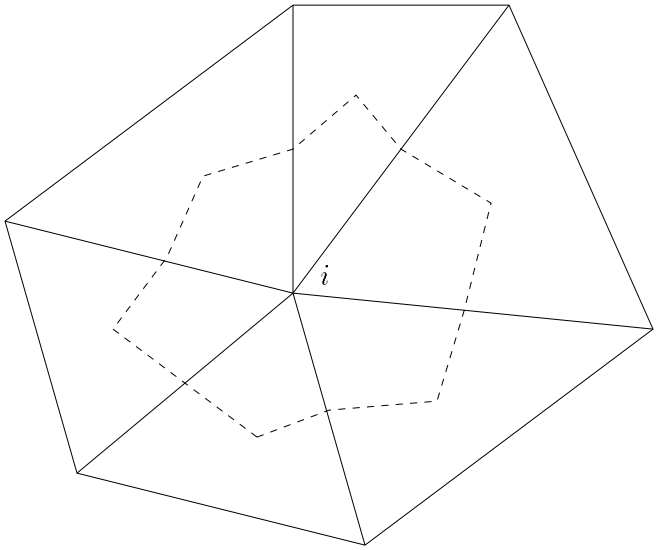

Figure 3.6: Secondary mesh element associated with an interior node

ing a finite difference expression into the time-derivative on the left-hand side and denoting B_{iz} as the boundary of V_{iz}, we can write the semi-implicit equation

$$V_{iz}\frac{u_{iz}^{n+1}-u_{iz}^{n}}{\Delta t}=\int_{B_{iz}} D^{n}\nabla u_{iz}^{n+1}.\underline{\hat{n}}dB_{iz} \tag{3.8}$$

where ∇u_{iz}^{n+1} is the linear approximation to the gradient function of u over its control volume, and $\underline{\hat{n}}$ is the outward normal vector out of the area V_{iz}. Again we use an explicit expression for the diffusion co-efficient D.

We now consider a section of the boundary to V_{iz} so as to derive an expression for these normal components. Consider the two adjacent triangles, $\triangle_{ikj}$ and $\triangle_{ijm}$ each containing the nodes i and j, which are connected by an edge defined by vector $\underline{j}$. To complete these triangles, we also define their remaining nodes as k and m, and the edges, ik and ij, connecting these to node i are defined by the vectors $\underline{k}$ and $\underline{m}$ respectively. Each of these nodes also belongs a contour which has its own respective current value u_{iz}^{n}. The section of the boundary we consider can be split into two edges. The first of these edges, connecting the centre of $\triangle_{ikj}$ to the midpoint of edge ij, is defined by the vector $\underline{b}_{ikj}$, while the second joins the midpoint of edge ij to the centre of $\triangle_{ijm}$ and is defined by the vector $\underline{b}_{ijm}$. These nodes, vectors and the resulting triangles are shown in Figure 3.7.

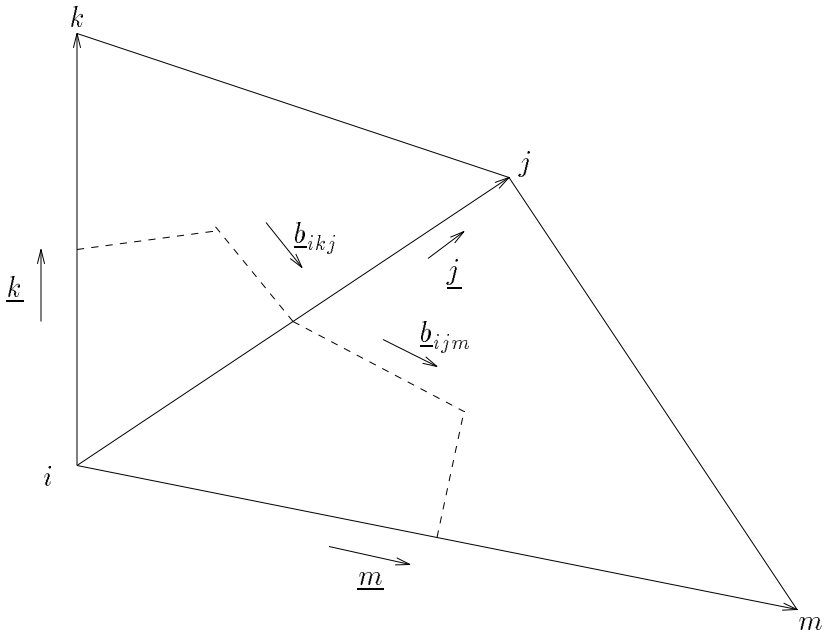

Figure 3.7: Nodes i,j and associated vectors.

We can now write

$$D^n \nabla u_{iz}^{n+1} . \underline{\hat{n}} = D_{ikj}^n \nabla u_{ikj}^{n+1} . \underline{\hat{n}}_{ikj} + D_{ijm}^n \nabla u_{ijm}^{n+1} . \underline{\hat{n}}_{ijm} \tag{3.9}$$

where $\underline{\hat{n}}_{ikj}$ is the outward normal of $\underline{b}_{ikj}$ and D_{ikj}^n is the value of D^n evaluated by linear interpolation at the centre of $\triangle_{ikj}$, and similarly for subscript ijm.

First consider the component $D_{ijm}^n \nabla u_{ikj}^{n+1} . \underline{\hat{n}}_{ikj}$. From [77] u is defined linearly over each triangle such that

$$u_{kz}^{n+1} = u_{iz}^{n+1} + \underline{k} . \nabla u_{ikj}^{n+1}$$

and

$$u_{jz}^{n+1} = u_{iz}^{n+1} + \underline{j} . \nabla u_{ikj}^{n+1}$$

and hence we may write

$$\nabla u_{ikj}^{n+1} = \frac{(u_{kz}^{n+1} - u_{iz}^{n+1}) \underline{\tilde{j}} - (u_{jz}^{n+1} - u_{iz}^{n+1}) \underline{\tilde{k}}}{\underline{k} . \underline{\tilde{j}}} \tag{3.10}$$

where $\underline{j}$ is vector $\underline{j}$ rotated clockwise by $\frac{\pi}{2}$.

We now have

$$D^n_{ikj}\nabla u^{n+1}_{ikj}.\underline{\hat{n}}_{ikj} = C_k u^{n+1}_{kz} - C_{j_1} u^{n+1}_{jz} + (C_{j_1} - C_k)u^{n+1}_{iz}$$

where

$$C_k = D^n_{ikj}\frac{\widetilde{\underline{j}}.\underline{\hat{b}}_{ikj}}{\underline{k}.\widetilde{\underline{j}}}$$

and

$$C_{j_1} = D^n_{ikj}\frac{\widetilde{\underline{k}}.\underline{\hat{b}}_{ikj}}{\underline{k}.\widetilde{\underline{j}}}$$

It can be shown from (3.10) that if nodes k and i belong to the same contour then $u^{n+1}_{kz} = u^{n+1}_{iz}$ and hence $C_k = 0$.

Following the same ideas for $\nabla u^{n+1}_{ijm}.\underline{\hat{n}}_{ijm}$, we have

$$D^n_{ijm}\nabla u^{n+1}_{ijm}.\underline{\hat{n}}_{ijm} = C_{j_2} u^{n+1}_{jz} - C_m u^{n+1}_{mz} + (C_m - C_{j_2}C_m)u^{n+1}_{iz}$$

where

$$C_m = D^n_{ijm}\frac{\widetilde{\underline{j}}.\underline{\hat{b}}_{ijm}}{\underline{j}.\widetilde{\underline{m}}}$$

and

$$C_{j_2} = D^n_{ijm}\frac{\widetilde{\underline{m}}.\underline{\hat{b}}_{ijm}}{\underline{j}.\widetilde{\underline{m}}}.$$

Again if nodes m and i lie on the same contour then $C_m = 0$. We now have a new expression for (3.9),

$$D^n\nabla u^{n+1}_{iz}.\underline{\hat{n}} = C_k u^{n+1}_{kz} + C_j u^{n+1}_{jz} - C_m u^{n+1}_{mz} + (C_m - C_j - C_k)u^{n+1}_{iz}$$

where $C_j = C_{j_2} - C_{j_1}$. Again if nodes i and j belong to the same contour then $C_j = 0$.

This gives an expression representing the contribution to the flow of material out of the iz^{th} contours control volume from node i to nodes k and m in their respective

contours kz and mz control volumes. The sum of these contributions from all nodes belonging to contour iz will be equal to the integral on the right-hand side of equation (3.8). So finally we have

$$V_{iz}\frac{u_{iz}^{n+1}-u_{iz}^{n}}{\Delta t}=\sum_{\forall i\in iz}\sum_{j}C_k u_{kz}^{n+1}+C_j u_{jz}^{n+1}-C_m u_{mz}^{n+1}+(C_m-C_j-C_k)u_{iz}^{n+1}. \quad (3.11)$$

The limits on the first summation denote all the nodes i belonging to contour iz, while the second summation denotes all the neighbouring nodes j of i. It is noted that this vector style formulation is notationally complicated, being taken directly from Winslow [77]. Later on in Chapter 7 we shall work with a more concise arrangement. When re-arranged, (3.11) leads to an $N \times N$ system, where N is the number of contours, stationary points and extra nodes on the grid, to be solved for the approximate value of u on the contour or node. In general this will lead to a one-dimensional tridiagonal system, as the nodes n,k and m can only belong to one of three contours, these being the iz^{th} contour itself or one of its two adjacent contours. Due to the relatively small dimensions of the resulting system and its structure, the equations can be solved easily and quickly. Again, since we are using the semi-implicit form of the diffusion co-efficient, non-linear problems can be tackled with a simple linear solver.

In line with the procedure presented above in Section 3.2, we choose an adaptive time step in exactly the same way. After choosing a time stepping parameter δu, an explicit approximation value of the solution at the local maximum u_{max} using the equation (3.11) is used. This results in the following expression for the adaptive time step length,

$$\Delta t=\frac{\delta u V_{maxz}}{|\Psi_{max}|}$$

where $\Psi_{max}=\sum_{\forall i\in iz}\sum_j C_k u_{kz}^n+C_j u_{jz}^n-C_m u_{mz}^n+(C_m-C_j-C_k)u_{maxz}^n$. Again, this choice of Δt can be supported by the invariance theory as in one dimension.

Having presented the equations for updating the contour heights, we now describe a method of moving the discretised contours.

3.3.3 Moving Contours

In Section 3.2.2, nodes were moved such that an equidistribution property was preserved at each time. Grid points were re-equidistributed over the updated solution via linear interpolation.

In two dimensions, the algorithm is unchanged, although a more complex interpolation process is now needed to find a set of nodes to represent a contour, rather than to locate a single point. Figure 3.8 shows two contours i and j with newly computed values, or 'heights' u_{iz}^n and u_{jz}^n. Now let us suppose that (due to monotonicity) a newly equidistributed contour level $\acute{u}_m^{n+1}$ lies between these two heights.

Linear interpolation takes place along all edges of the grid connecting nodes belonging to contours i and j, to find the points along these edges which have the prescribed solution value $\acute{u}_m^{n+1}$. We now have a discretisation of the newly placed contour denoted by the broken line. However it is noticed that the new contour is defined on nearly twice as many edges as desired points. So that the new contour has the correct number of nodes such that connectivities throughout the mesh remain constant, we have to interpolate again, this time along u_m^{n+1}, which is done in such a way that nodes are again equally spaced over the contour.

We have now described the moving grid method in two-dimensions. The algorithm is implemented in exactly the same way as in one dimension (see Section 3.2.3). The following chapter shows results for two problems of non-linear diffusion type in both one and two dimensions and analyses and assesses the approximate solutions.

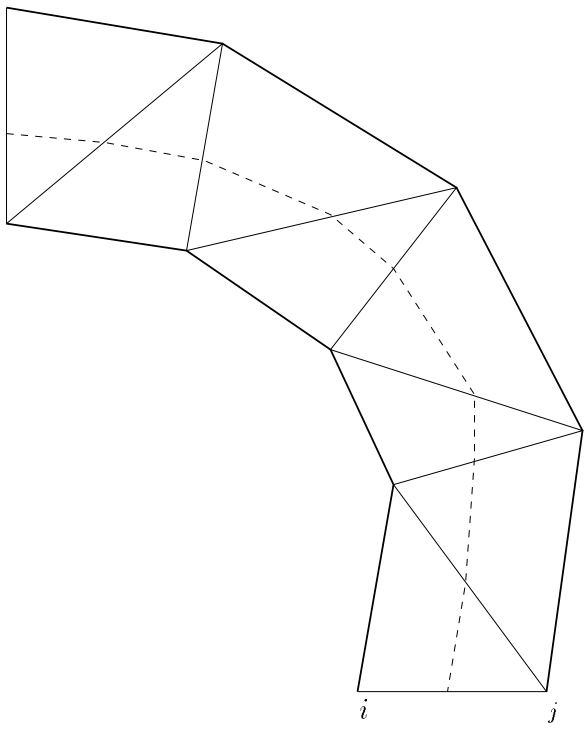

Figure 3.8: Defining a new contour in Two-Dimensions

Chapter 4

Numerical Results I

We now present numerical solutions using the methods described in Chapter 3. We test the algorithm using two problems of non-linear diffusion form as highlighted in the derivations. The first application is closely linked to the PME. Hereafter referred to as the semi-conductor problem, a modified non-linear diffusion coefficient is used to simulate the diffusion of dopant within a crystalline silicon semi-conductor device upon being heated. The first problem is the PME mentioned in Chapter 2.3. In the first section we use the regridded contour zoning solution to generate approximate solutions to the two problems in one dimension. Technical details regarding the implementation of the methods are also included along with full statement of the two problems. Section 4.2 uses the full contour zoning method to solve the higher-dimensional versions of the two problems.

4.1 One Dimension

The results presented in the two following sections were generated in MATLAB using a Gaussian Elimination routine to solve the linear system of equations arising from (3.3). We first consider the semi-conductor problem.

4.1.1 Semi-Conductor Problem in One-Dimension

As mentioned above, this problem arises in semi-conductor process modelling, where a dopant is introduced and diffused within a silicon device. The distribution of the

dopant is needed to produce certain electrical properties of the semiconductor itself. The dopant is diffused through the silicon when the device is heated, the diffusion process itself is non-linear, and the evolution is modelled by the equation

$$u_t = ((u + \epsilon)u_x)_x.$$

The doping material is introduced into the silicon by ion implantation through the silicon surface and results in a high concentration of dopant in a shallow region. The initial conditions take the form of a Gaussian function centred about the origin. This initial doping is done en masse, with many such amounts of material equally spaced over a large area of silicon (see Figure 4.1). However, for ease of illustration we choose to work with one such section, taking one half of the initial distribution centred at $x = 0$, over an arbitrary domain $0 \leq x \leq 1$. The Gaussian function takes the form

$$u(x, 0) = e^{-\sigma x^2}.$$

Neumann boundary conditions are imposed at both boundaries, representing the conservation of mass and the symmetry of the initial conditions. Hence we have that

$$u_x(0, t) = u_x(1, t) = 0.$$

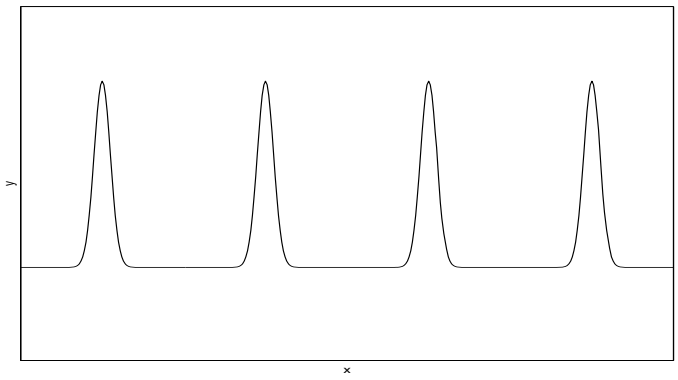

Figure 4.1: Initial distribution of dopant en masse over semiconductor

The distribution of dopant over time has previously been solved numerically, both by following a suitable variable transformation by Please and Sweby [66] and

by using moving finite elements by Hobbs [42]. King & Please [53] also considered the problem using an asymptotic approach.

Preliminary results are shown in Figure 4.2. The approximate solutions are generated using 25 nodes, each solution value being represented by the cross hairs in the figure. Intermediate results are output at the re-equidistributed stage, and the equally spaced values over the solution range, characteristic of the monitor used, are easy to spot. The parameter ϵ is set at a value of 0.01 in line with the work of Hobbs [42] and the initial conditions are generated with $\sigma = 50$. Finally the time-stepping parameters δu and Δt_{max} are taken to be 0.005 and 0.01 respectively.

The dopant diffuses as expected. Initially the diffusion is fast, the final two graphs demonstrating how this motion slows down considerably as time progresses. Figure 4.3 supports this idea of change in motion and the effectiveness of the time-step in adapting to the rate of change in activity. The left hand side of the figure shows how the time step behaves with time, with the numerical time integration of the equation initially being very slow in order to cope with the rapid changes in u at the start of the computation. Eventually the computation picks up pace as the activity dies down and finally the linear features of the graph indicate when the maximum time step has started to take effect. The right hand side of the figure shows the trajectories of the nodes. Again the nodes move rapidly to adjust to the initial pace of the diffusion. With time the nodes move with less vigour and even move backwards as the solution settles. In the later stages of the computation, the motion of the dopant settles down and the nodes become more or less evenly spaced over the domain.

Despite the success of the adaptive time step used, a major flaw can be noticed in the initial representation of the solution. Instead of the expected tail of such a Gaussian function, the positioning of the nodes instead gives the Gaussian a shallow ramp. This implies that the solution is working from a perturbed initial state. In turn, this implies that the total mass of the dopant is incorrect. More importantly, this will affect the diffusion of the dopant in regions of high concentration. Of particular interest in this problem is the amount of dopant in areas away form the main body of material. The diffusion coefficients at the tail region will be

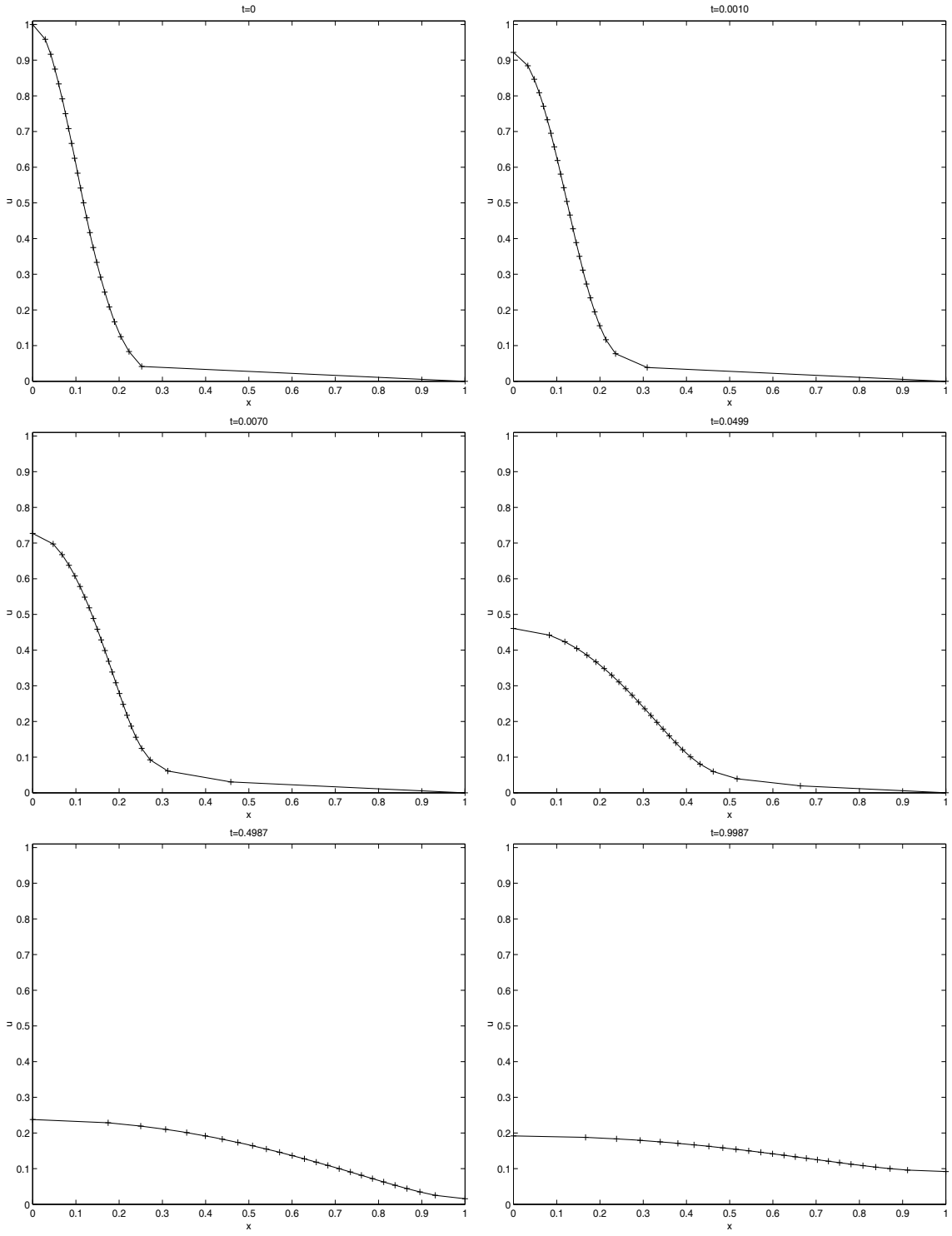

Figure 4.2: Approximate solution to Semi Conductor Problem using 25 nodes, with $\epsilon = 0.01$.

greater than required and hence the amount of material in the region near to $x = 1$ approximated in the long term solution will be greatly over estimated. However the following section outlines a modification to the method to help us have a more accurate representation of the initial distribution of the dopant.

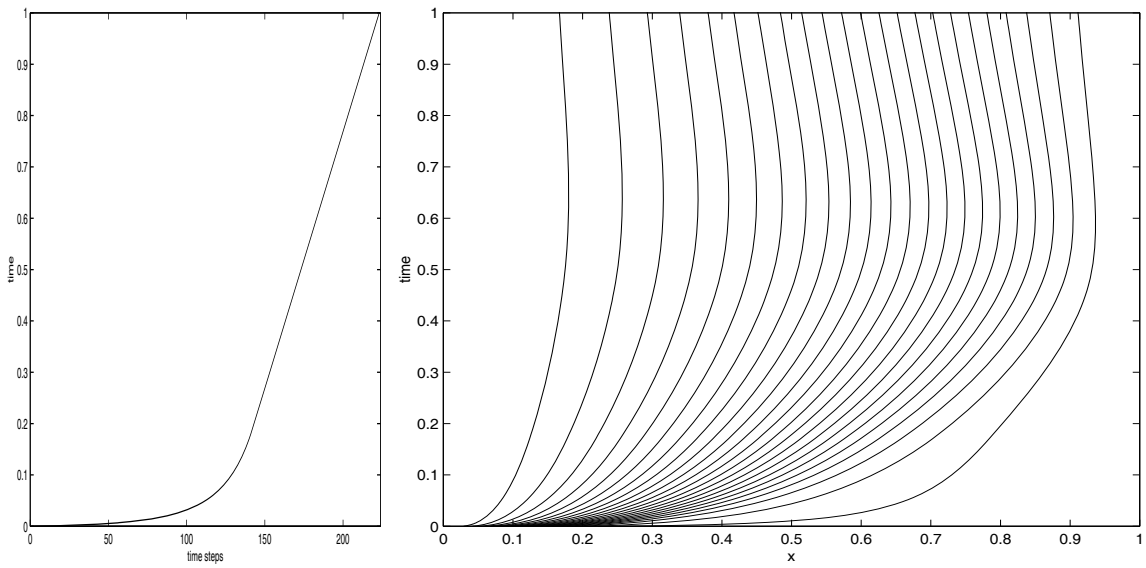

Figure 4.3: (Left) Development of adaptive time step. (Right) Trajectories of nodes with time.

4.1.2 Grid Refinement and Regridding

The aim of this section is to adapt the way in which we choose the heights of the contours in the relocation of grid points, in order to give a greater resolution in the tail region of the solution.

The ease of choosing the heights of contours gives us a reasonable amount of freedom in redistributing grid points over the domain. There are any number of ways of choosing these heights. We could, for instance, simply divide up the solution range equally in the last cell to accommodate an extra M nodes within the last cell. However we would like to distribute the grid points smoothly over the solution range.

Assuming an initial number of nodes $N + 1$ and a specified number of 'extra' nodes, M, to be added to areas where the representation of the solution is poor. By calculating an increment over the solution range, γ, we distribute the extra nodes in the tail region such that the first additional node away from the right

hand boundary has the value $u_{min} + \gamma$, the second has value $u_{min} + 2\gamma$ and so on, until all the refinement nodes have been assigned values. The initial nodes are then separated by $(M + 1)\gamma$. To further enhance the representation of the Gaussian, an equal number of nodes are added in the first cell at the peak of the function. This should further enhance the calculation of the adaptive step size since the grid resolution will be guaranteed to be finer near $x = 0$. This *ad hoc* refinement process, although no longer satisfying any global equidistribution idea, is easy to implement and the only change in the solution algorithm is the calculation of the new values of the grid points to be redistributed.

We now put the procedure into practice using the same values for σ and ϵ as in the previous computation. We choose the values $N = 12$ and $M = 6$ so that the total number of nodes is the same as that used in Section (4.1.1). The corresponding results are shown in Figure 4.4.

It is easy to see that the ramp has been removed from the initial conditions. Although the resulting solutions at first glance do not seem to differ greatly from those presented earlier, it is clear that a more accurate approximation to the total mass of dopant is present. In the early development of the solution a much smoother approximation is obtained. This has a subtle knock-on effect for the long-term solution in that a lower percentage of the mass has diffused to the further reaches of the domain. The final state at $t = 1$ shows the difference in total material present in the solution.

We have not discussed the effect of the interpolation used in redistributing grid points. The effect is clearly illustrated when looking at the total amount of dopant present in the computation via linear quadrature. The scheme used in updating the solution, as presented in the previous chapter, is conservative with the application of the correct boundary conditions. However this property is lost when the solution is interpolated between time steps. Figure 4.5 shows the total mass as approximated via a composite trapezium rule with time. For reference, a fine-scale numerical approximation to the total mass involving the error function is plotted along with the approximation to the integral with time for various values of N and M. The approximations of total mass are plotted first when no refinement is used with 25

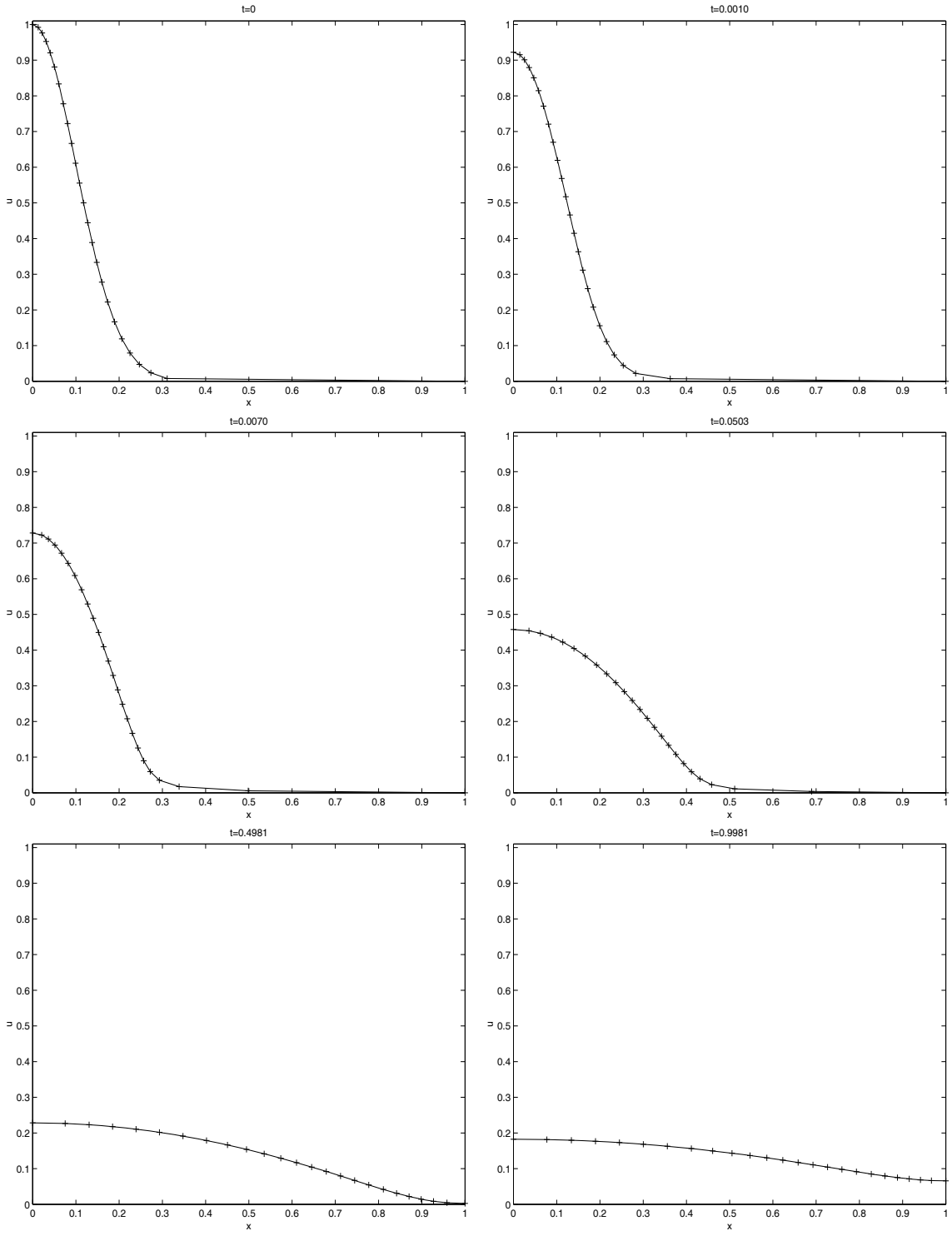

Figure 4.4: Approximate solution to Semi Conductor Problem with refinement using 25 nodes with M=6 and $\epsilon = 0.01$.

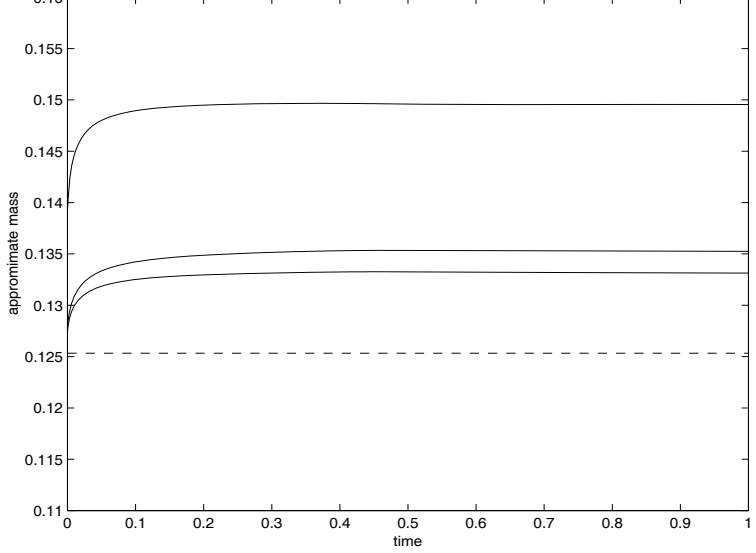

Figure 4.5: Linear approximations to mass with time, using no refinement, M=6 and M=10. Exact Mass also shown

nodes, then with values of $M = 6$ and 10, keeping the total number of nodes as 25. As seen when refinement is used, the initial approximate quantity of dopant is more accurate than with no refinement. However in all cases the mass grows as interpolation is used to transfer the solution between stages of regridding. Further evidence that interpolation error is the cause for this increase is the fact that the mass grows rapidly in early stage of the evolution of the solution. This is significant since in these stages the time step is smaller, so more time steps are taken and hence more interpolation error is induced.

4.1.3 Porous Medium Equation in One-Dimension

We begin by re-iterating the PME in one-dimension,

$$u_t = (u^m u_x)_x,$$

where $m > 0$.

We have already noted the existence of an analytic solution (2.38) arising from an insect dispersal model found in Murray [64]. Since the initial conditions used in the model ($u = Q$ at $x = 0$, $u = 0$ elsewhere) are not easily represented, we choose

an arbitrary time ($tstart > 0$) to begin our computations, the corresponding state of u being given by the Murray solution (2.38).

Since the equation conserves both mass and centre of mass, the symmetry of the solution about $x = 0$ allows us to concentrate on only the positive half of the problem.

We now must deal with the moving boundary involved in the solution of the PME. We have a conflict of interests since the contour zoning method is essentially a static moving mesh method and hence does not strictly involve speeds of mesh points. In order for the regridding procedure to allow the moving front to progress we must have a node fixed outside the region of compact support and choose suitable conditions such that minimal flow will be allowed out of the moving region and in some sense reflect the properties of the solution.

Since conservation of mass powers the movement of the front, it is essential our scheme preserves this property. Fixing a node $x_N = 1$, the natural choice is to impose Neumann conditions at each boundary. This will also preserve the symmetry of the solution about $x = 0$ (i.e. conservation of the centre of mass). It would be advantageous to impose a fixed value of $u = 0$ at x_N. However the current system of equations used for updating the solution will not conserve mass with this additional condition. To try to prevent as little material as possible seeping out of the moving region of compact support, we will need to use the refinement algorithm implemented in the semi-conductor problem. So our boundary conditions are

$$u_x(0,t) = u_x(1,t) = 0$$

Moreover, this implies that the numerical solution will become invalid as the penultimate node moves towards $x = 1$. So we shall only integrate as far as the time suggested by the Murray solution as being when the front reaches the far boundary.

We consider the three integer values of $m = 1, 2, 3$. As explained in Section 2.3, as the value of m increases so too does the steepness of the moving front. Furthermore, variations in m affect the speed of the front. As m is increased the variation in speed initially is increased, i.e. the deceleration of the moving boundary is increased with m. Figures 4.6 and 4.8 show the approximate and analytical solutions for the various

values of m for the times at which our solution is valid.

In all three cases, the regridding algorithm performs considerably well at the left hand boundary. However it is at the moving boundary where differences between the numerical and Murray solution appear. Since we cannot impose the Neumann condition at the foot of the travelling front, a certain amount of diffusion has occurred outside of the intended region of compact support. Initially the grid refinement does well to contain the diffusion. This works in two ways. Firstly, since the values of u at these refinement points are generally small and the diffusion coefficient is a power of u, then the rate of diffusion at these points is minimal. Secondly, for the larger values of m, these nodes provide a more accurate resolution to the steep developing front. However at later times the diffusion at these points builds up and the analytical and approximate solutions become more and more distant from one another.

The existence of the Murray solution gives use a useful tool with which to measure the accuracy of the method, although many error measures may give confusing results in this case due to the existence of the moving boundary in the problem. For this reason we use a simple average error measure, i.e. sum of errors at all points in the mesh divided by the number of nodes. Figure 4.9 plots the log of this error measure against the number of points used to check for an order of accuracy. In order to control all parameters, for this experiment we dispensed with the adaptive time step and used a constant one to capture the effect of additional points on the solution. As with previous results in Blake [9] the graph suggests the regridding procedure has first order accuracy.

4.2 Contour Zoning in Two-Dimensions

We now implement the full contour zoning solution algorithm in two dimensions. In the previous section it was shown that using the grid refinement procedure would be beneficial to both the semi-conductor and the porous medium problems. Hence it is intuitive to use the same ideas in two dimensions. The only change to the procedure outlined in Chapter 3 is the way in which the heights of the contours on

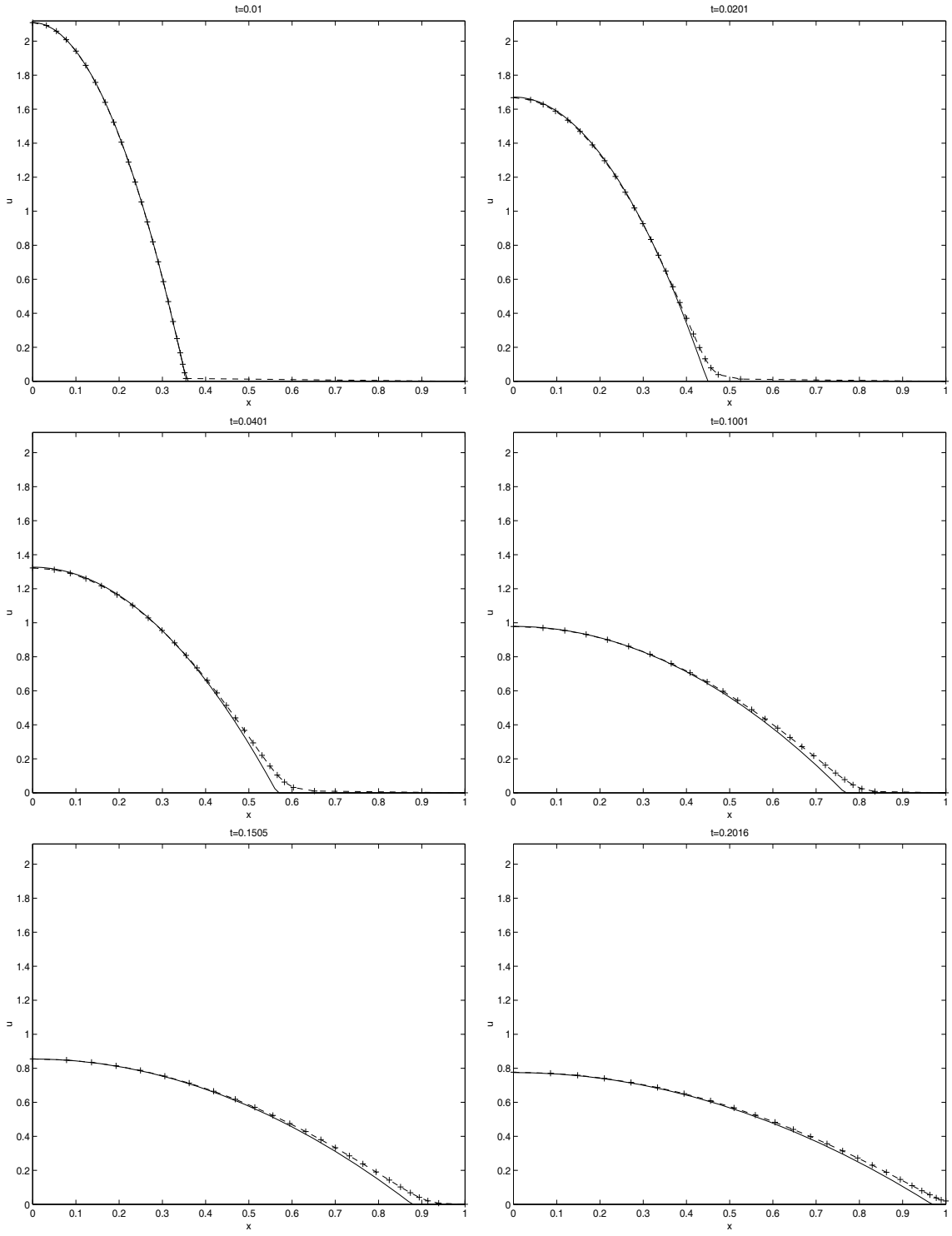

Figure 4.6: Approximate solution to the PME with refinement using 25 nodes with M=6 and $m = 1$.

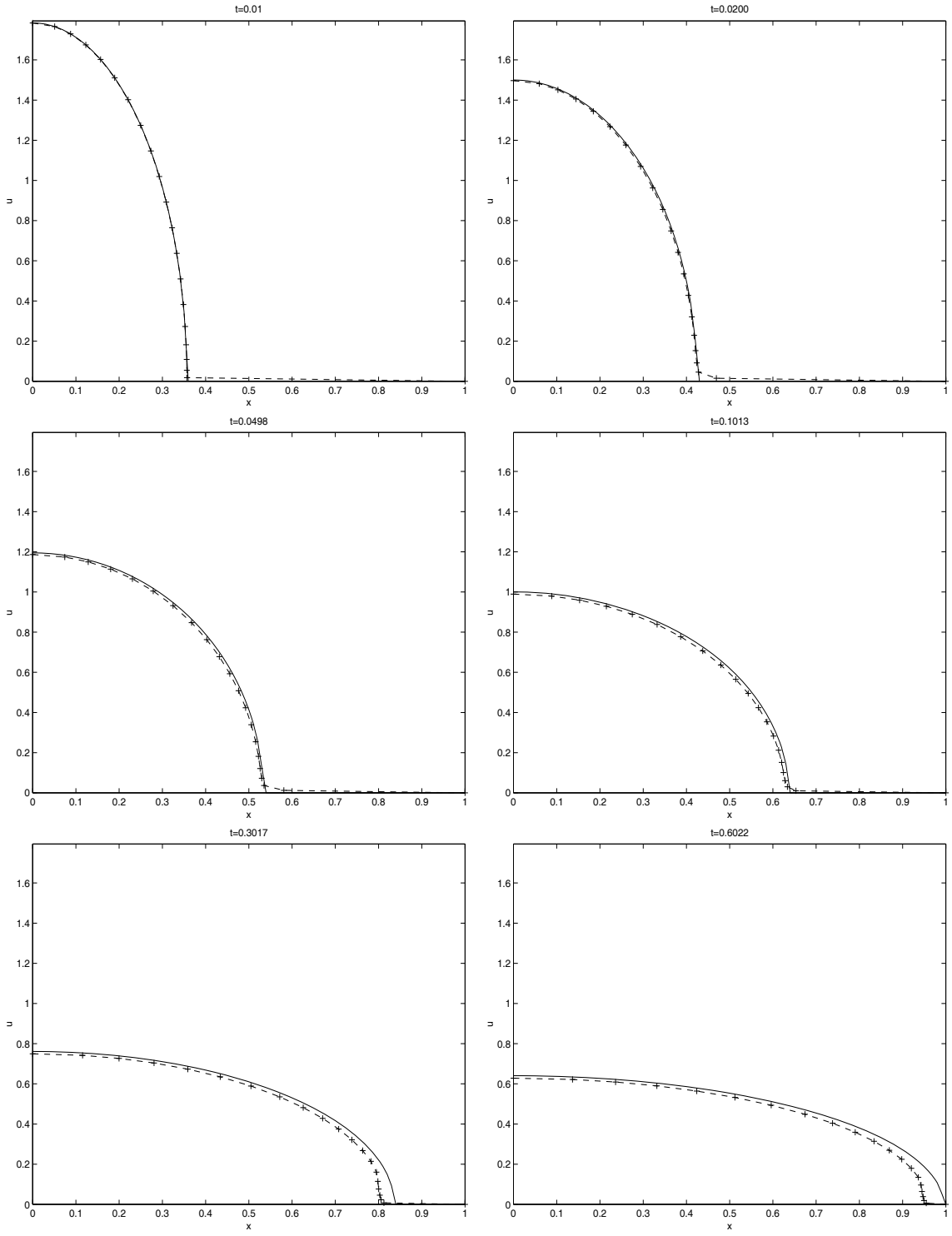

Figure 4.7: Approximate solution to the PME with refinement using 25 nodes with M=6 and $m = 2$.

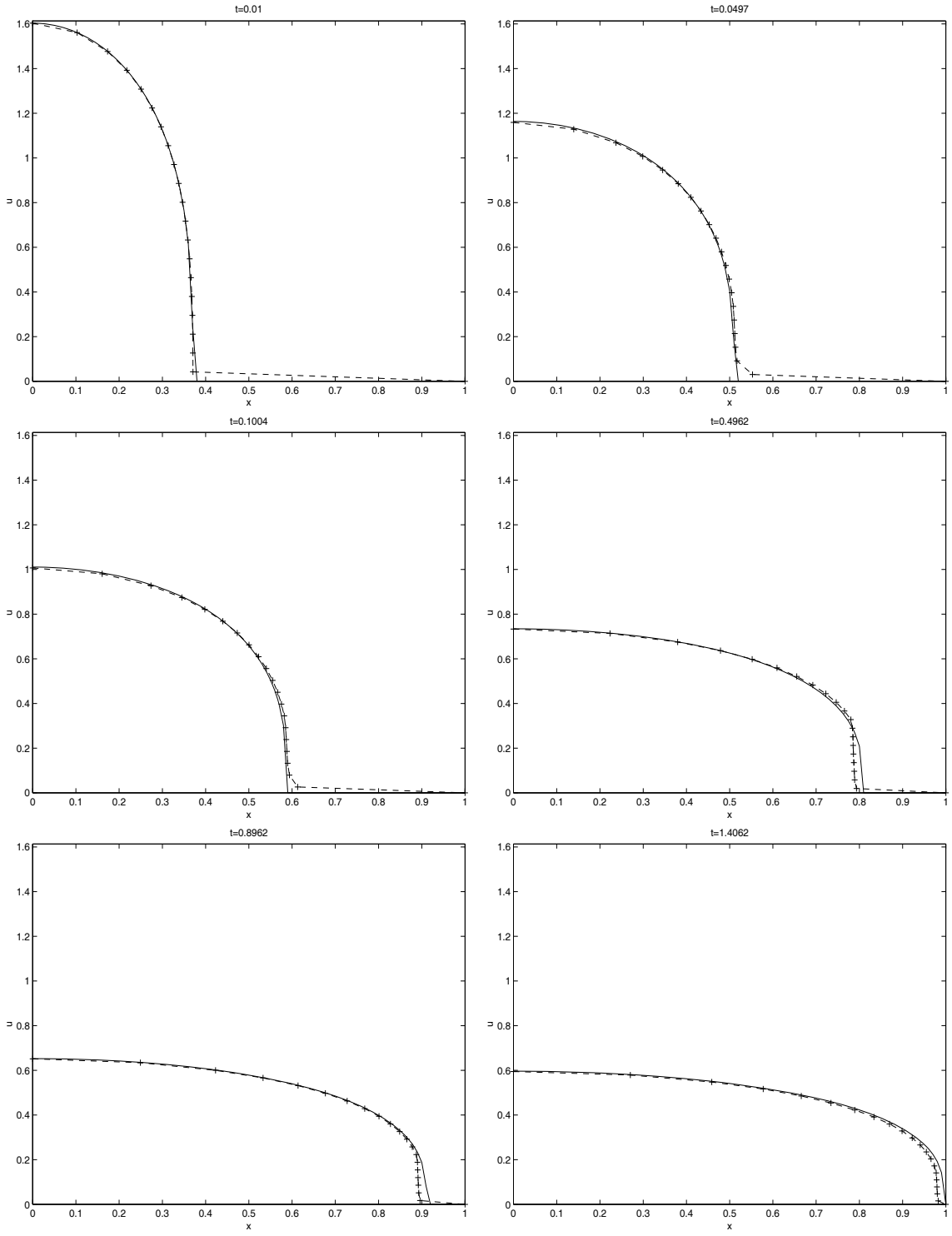

Figure 4.8: Approximate solution to the PME with refinement using 21 nodes with M=1 and $m = 3$.

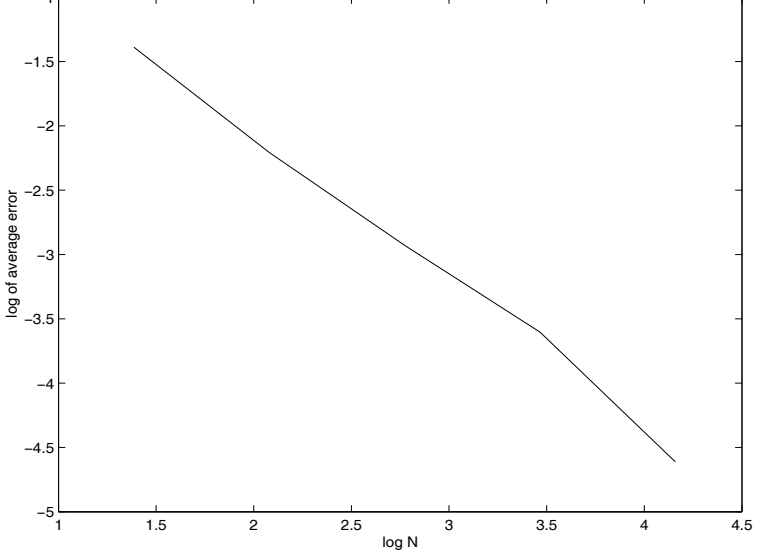

Figure 4.9: Error convergence with N, $m = 1$.

the newly reconfigured grid are chosen. For continuity we choose these heights in the same way as described and used in Section 4.1.2. The initial computational grids were constructed using MATLAB, with the contour zoning solution carried out in a FORTRAN90 code, as with the one-dimensional work. The equations for the values of u on each contour were solved using a Gaussian elimination style tri-diagonal solver.

4.2.1 Semi-Conductor Problem in Two-Dimensions

We begin by considering the two-dimensional analogue of the semi-conductor problem. It is assumed now that the Gaussian conditions for the initial distribution of the dopant are non-radially symmetric. We take an arbitrary Gaussian hump such that

$$u(x, y, 0) = e^{-(8x^2+100y^2)}.$$

We now need a fixed boundary for our domain. In two dimensions the solution has origin $(0, 0)$ and has reflective symmetry in both axes. For ease in constructing the initial grid for these conditions, we choose our domain to be bounded by the positive x and y axes and a particular contour corresponding to an arbitrarily chosen

'minimum' height. For our computations this height is chosen as 10^{-11} and Neumann conditions are imposed at all of the boundaries to conserve mass and maintain the symmetric properties of the model.

Figure 4.11 shows plots of the initial, an intermediate, and the final ($t = 1.0$) grid and solution for the semi-conductor problem. Using similar parameters as in the one-dimensional calculations, the grid was constructed with a total of 21 contours including 5 refinement contours at the upper and lower sections of the solution range. Furthermore 15 nodes were used to discretise each contour giving a grid comprising of 301 nodes and 546 triangles. With regard to the remaining solution parameters, δu and ϵ were set at 0.005 and 0.01 respectively.

The solution diffuses in much the same way as in the one-dimensional case, with an initial rapid burst and then a slower rate over the chosen domain. We expect the same effects of interpolation error as encountered in one-dimension, indeed we could expect more error to be introduced in this way since interpolation is now done over two, rather than one, dimension. However, Figure 4.10 confirms that the use of the extra contours improves the initial approximate mass conserved in the problem.

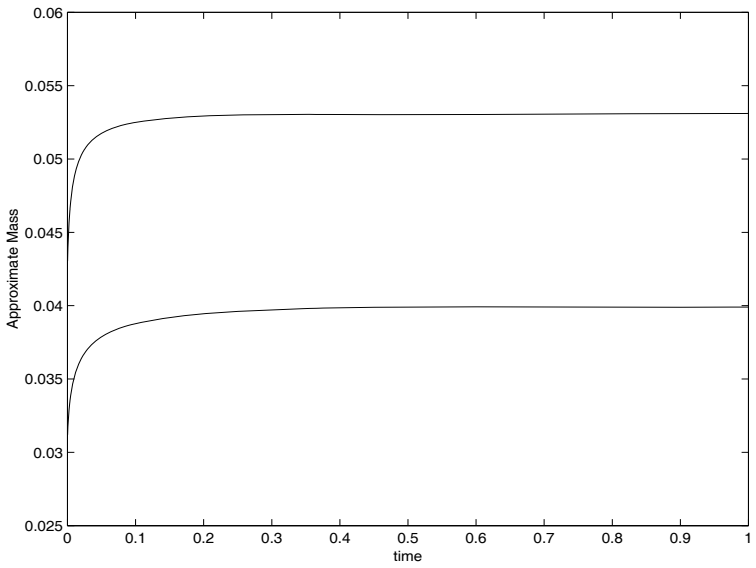

Figure 4.10: 2D Linear approximations to mass with time, using no refinement and M=5

Notice however that the contours seem to propagate outwards uniformly. Since our initial conditions are non-radially symmetric, then gradients in directions normal

to the contours will not be equal over the contour. Thus we would expect faster diffusion in certain parts of the contours than others. This would then mean that in some regions contours would contract together and in other areas move away from each other. The contour-zoning solution method has not been able to resolve these features. Close inspection of the algorithm reveals that the actual contour acts as a central point for the sum of associated control volumes of the nodes on the contour. Hence in a way the process of collecting all fluxes out of and into this conglomerate volume may actually average the effect of the individual fluxes, in return for having only one equation to solve for the height of the contour. Hence we cannot expect the contour zoning procedure in conjunction with this regridding procedure to provide enough freedom for the possibility of contours moving with changing geometries.

We finally mention the cost of implementing the method. The computations presented in Figure 4.11 were undertaken on a SUN ULTRA5 workstation. The adaptive time stepping procedure used a total of 254 time steps for the time integration, each step taking an average of 0.008557 seconds to complete.

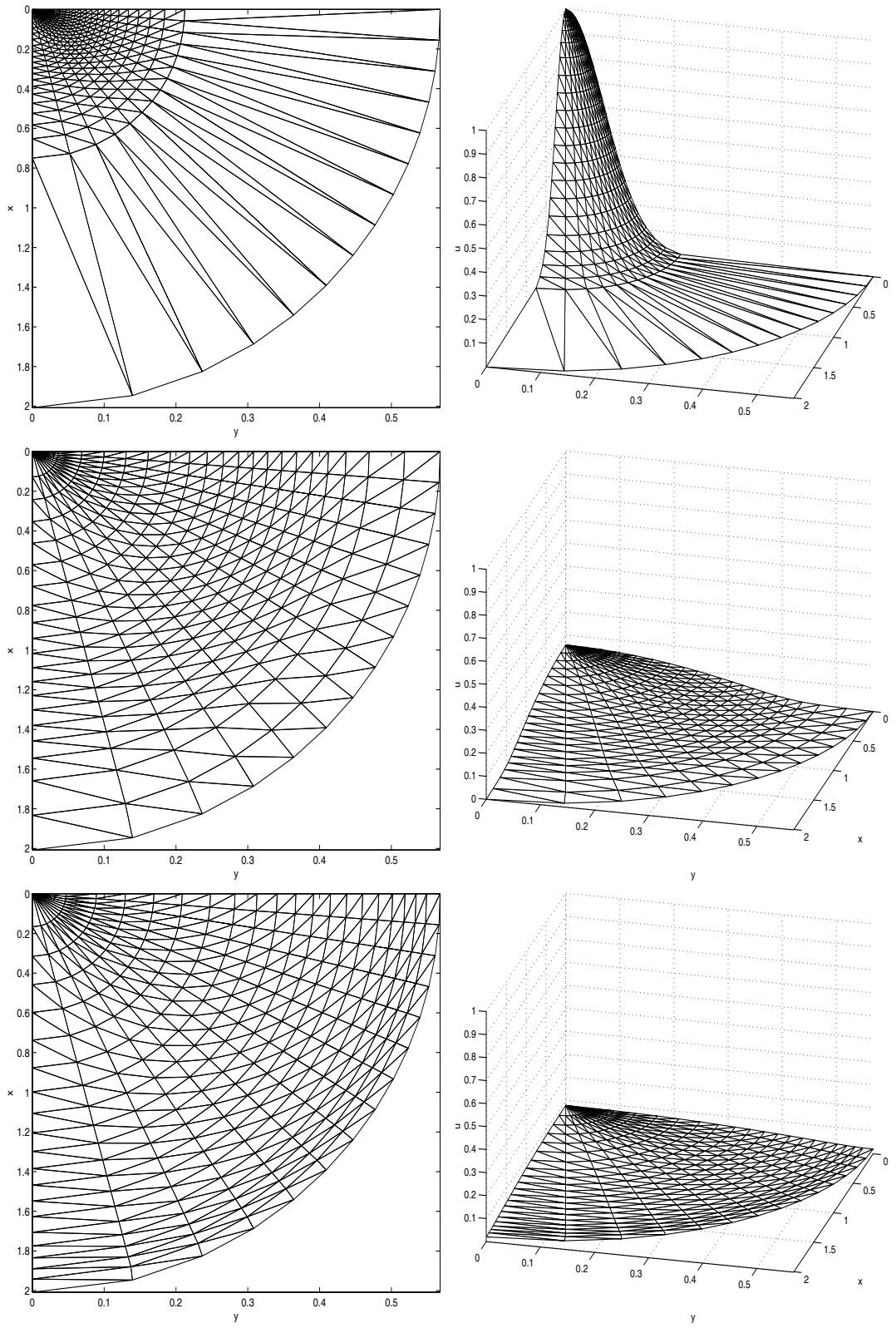

Figure 4.11: Grids and solution to semi-conductor problem in 2D. Initially (above), at $t = 0.2$ and at $t = 1.0$ (below).

4.3 Porous Medium Equation in Two-Dimensions

We now implement the contour zoning solution in two dimensions, using the radial symmetric version of the Murray solution (2.40) in the same way as in the one-dimensional case. We choose a maximum radius of $r = 1$ at which to impose the Neumann conditions and choose an arbitrary value to start our computations. To demonstrate how the contours are distributed along the steep front we choose $m = 3$. It is worth noting that the radial form of the solution is used for comparison only, our actual computations being derived from the full higher dimensional PME, without using the geometry of the radial case.

Figure 4.12 shows the grid and solution initially and at a final time of $t = 0.5$. Notice how the contours have aligned themselves well on the moving front. Moreover the cross sections of the solutions shown in Figure 4.13 compare well to the analytic solutions and illustrate a reasonable match allowing for the diffusion occurring beyond the moving front, as expected from the one-dimensional results.

Figure 4.14 shows the error measure with time at the origin. As found in Section 4.1.3, the method seems to give a good level of accuracy at the maximum. With most unsteady problems, you would expect a numerical scheme to accumulate errors with time, but during our computations the error remains below 10^{-2} despite the extra interpolation error introduced. The method may also suffer since the approximation to the expanding front region becomes less accurate in terms of spatial resolution along the contours, since we keep the number of points on the contour equal throughout time in order to keep the connectivity of the mesh constant.

4.4 Remarks

We have presented numerical results generated by the methods outlined in Chapter 3. The Contour Zoning approach in two dimensions becomes a strict regridding method in one dimension with nodes being relocated around the computational mesh via the use of the monitor $M(u) = u_x$.

In one dimension the method performs reasonably well, despite several shortcomings. The semi-conductor problems highlights the problem of error induced via

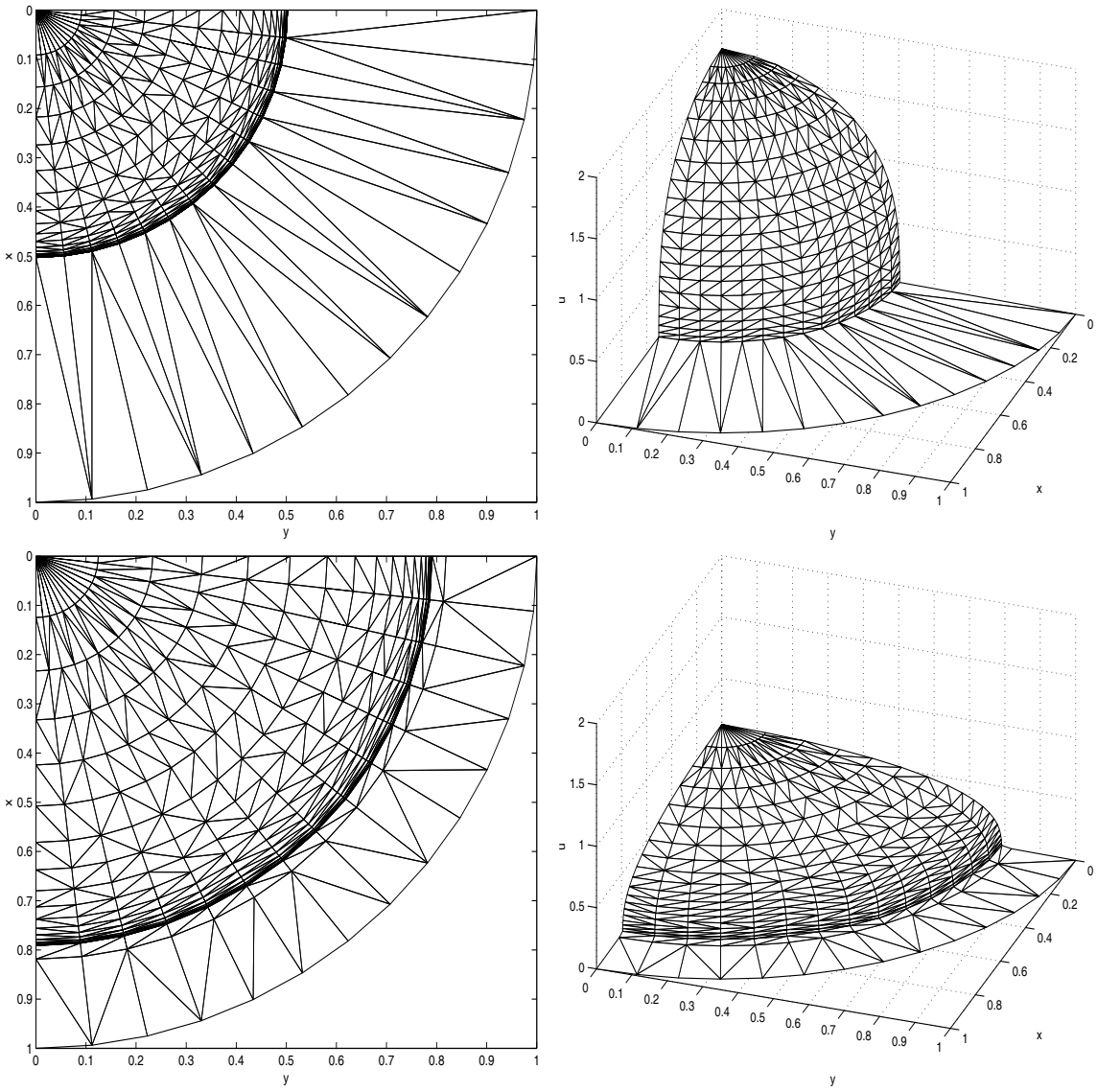

Figure 4.12: Grids and solution to the PME in 2D. Initially (above) at $t = 0.01$ and at $t = 0.5005$ (below) with $m = 3$.

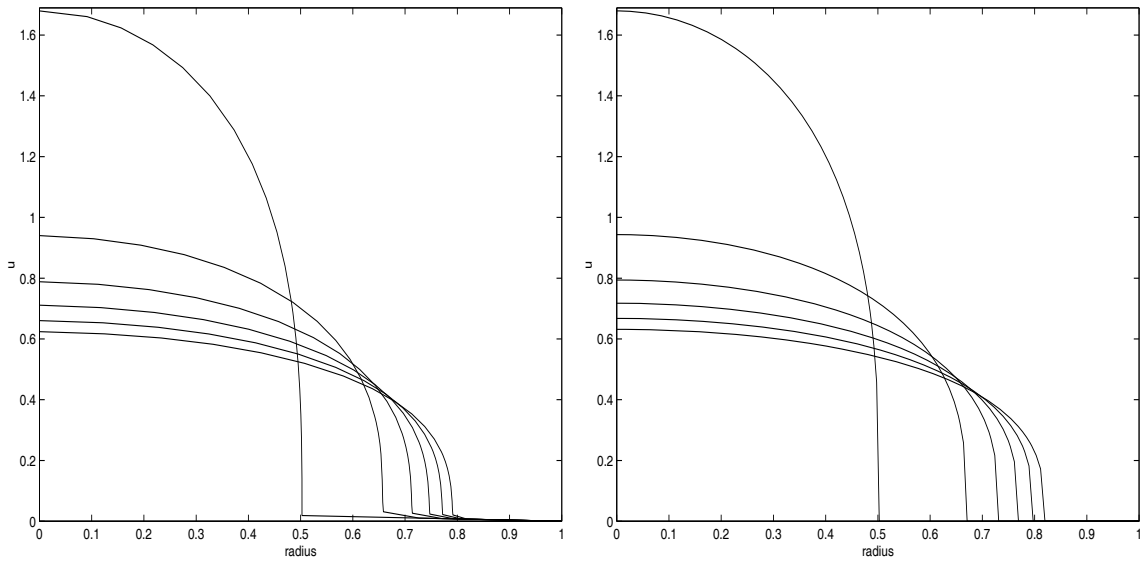

Figure 4.13: Cross sections of the approximate (Left) and the exact (Right) solutions at $t = 0.01, 0.1005, 0.2005, 0.3005, 0.4005, 0.5005$

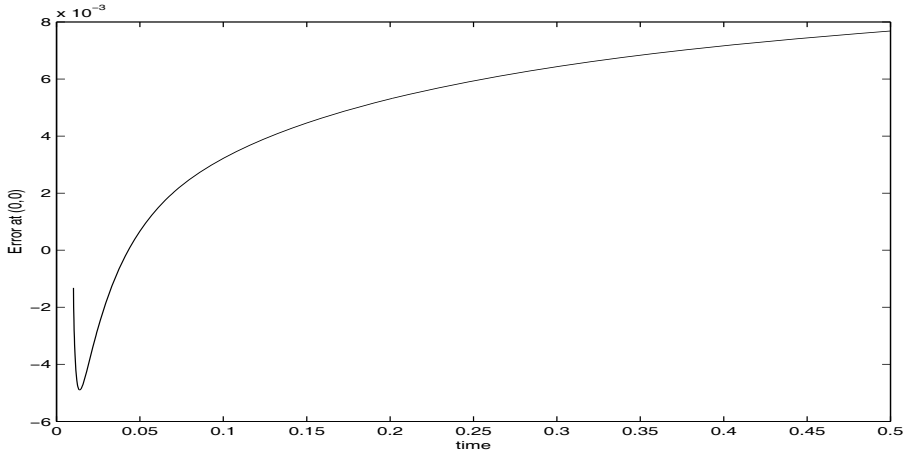

Figure 4.14: Error in contour zoning solution at origin.

the interpolation step from the old to new states of the mesh. This has implications on the total mass contained within the solution, which should, by a property of the problem, be conserved. Hence the problem is in a sense disturbed at each time step to a marginally different distribution. Although we have slightly improved this problem by the use of grid refinement, it would be beneficial to use a method which conserves mass. An obvious choice would be to use a mass-conserving interpolation method, but these usually involve strict conditions on local maximum and minimum, which do not agree with the problem in hand. Another idea would be to only undertake the regridding step at certain intervals in the time integration, minimising the increase in mass caused by the interpolation. This approach has been highlighted earlier in the work of Petzold [34] and results have suggested why this approach would be more beneficial. When considering the solution of the PME, we also had problems dealing with the moving boundary. In particular the method did not permit strict implementation of the boundary conditions, which led to poor approximations as the solution developed. These problems extended with the method into two dimensions too.

Another problem in two dimensions, highlighted by the semi-conductor case, is the apparent inability of the contour zoning method to be able to change the geometric shape of the contours with time. However, the method is suited to simple propagation of the contours and hence well suited to the radial case of the PME.

We conclude this section with some remarks regarding how to improve the solution of the two problems. In the semi-conductor problem, the first difficulty lies in correctly approximating and conserving the mass in the solution. The integration of u in time is done conservatively in both one and two dimensions, but perturbations to the mass are induced via the interpolation process used when relocating nodes and points. Secondly, in two dimensions the need for the contours to be able to change shape is noted. Despite the savings gained by allocated a single value over the contour, the way in which the contours are moved doesn't easily permit geometric changes. For a more accurate moving grid solution, nodes may have to be allowed to move more independently.

In the case of the PME, mass conservation is also a concern since this is the

primary source of the movement of the boundary. For a more accurate solution and to preserve mass inside the region of compact support, we need a moving mesh method that incorporates a moving or expanding domain. With this in mind, the next chapter introduces a more dynamic moving mesh approach, which not only accommodates the moving boundary, but also ensures mass conservation by construction.

Chapter 5

A Moving Mesh Method in One-Dimension

The results presented in the previous chapter highlighted some shortcomings of the contour zoning or regridding approach. The main disadvantages of the algorithm were the inability to conserve mass and to deal appropriately with a moving boundary problem. The former is of most importance especially since both our test problems conserve mass, while a satisfactory approach to the latter would allow us to start with a more solid foundation with which to compute effective numerical solutions to the PME.

In this chapter we present a moving mesh method to take care of these difficulties. The method stems from observations on equidistribution and geometric conservation laws from various papers from the literature covered in Chapter 2. In the following sections we shall follow the development of the method from its initial state equidistributing integrals of mass through to the introduction of more complex monitor functions. All stages of the development of the algorithm are supplemented and illustrated with numerical results. The method is presented first in one-dimension and with explicit reference to the PME. Later, in Chapter 6, we use our findings and return to compute solutions to the semiconductor problem and a problem involving blow-up using the new moving mesh routine. We begin by deriving a moving mesh method, driven by the conservation and equidistribution of mass.

5.1 Mass Conservation and Equidistribution

As we have seen from the literature covered in Chapter 2, the choice of monitor function can be crucial in the solution of a specific problem. For instance, Qiu & Sloan [67] developed a specialist monitor (2.26) for the solution of a reaction diffusion type problem where more traditional monitors failed. Of particular interest when considering the PME is the work of Budd et al ([15], [16], [18]) whose choice of monitor was heavily influenced by the theoretical invariance properties of the underlying PDE. In particular the mass monitor, $M(u) = u$, was singled out as a sensible choice for the PME since it too was invariant under scaling and would conserve mass.

Seizing upon this, it is noted that by choosing this monitor we can derive a moving mesh method without the need for the use of a computational or reference grid. The resulting mesh x and solution u can then be coupled in such a way that only the mesh needs to be integrated forward in time, with the solution being recovered from the current grid positions together with an integral quantity relating to the chosen monitor.

5.1.1 A Moving Mesh Equation

We begin with a simple equidistribution principle, working on a grid comprising of $N+1$ nodes $x_0, \ldots, x_N$, and, using notation in line with the existing literature, we have that

$$\int_{x_i(t)}^{x_{i+1}(t)} u dx = \theta(t) = \frac{1}{N} \int_{x_0(t)}^{x_N(t)} u dx \qquad i = 0, \ldots, N-1. \tag{5.1}$$

As previously noted, many moving mesh methods are derived from introducing node speeds into existing grid adaption statements. When considering the PME with its mass conservation property, direct time differentiation of the above equidistribution rule (5.1) leaves us with the simple expression,

$$\int_{x_i(t)}^{x_{i+1}(t)} u_t dx + u(x_{i+1}, t)\dot{x}_{i+1} - u(x_i, t)\dot{x}_i = \theta_t. \tag{5.2}$$

Substituting in the PME (2.30) and simplifying the integral term on the left-hand side leaves us with the moving mesh equation,

$$[u^m u_x]_{x_i(t)}^{x_{i+1}(t)} + u(x_{i+1},t)\dot{x}_{i+1} - u(x_i,t)\dot{x}_i = 0. \tag{5.3}$$

It is crucial to note, with this specific choice of monitor $M(u) = u$, that θ becomes independent of time since the PME is mass conserving. Hence, the zero right hand side in (5.3). Moreover, due to the symmetry of our porous medium solution, we have that $\dot{x}_0 = x_0 = 0$, so the above equation, when rearranged, leads to the sequence of ordinary differential systems (ODEs) for the grid co-ordinates,

$$\begin{aligned} \dot{x}_0 &= 0, \\ \dot{x}_i &= \frac{1}{u_i}\left(u_{i-1}\dot{x}_{i-1} - [u^m u_x]_{x_{i-1}(t)}^{x_i(t)}\right) \qquad i = 1,\ldots,N. \end{aligned} \tag{5.4}$$

To discretise the system we use an upwinding approximation for the space derivative terms, i.e.

$$u^m u_x|_{x_i} \approx \left[\frac{u_i + u_{i-1}}{2}\right]^m \frac{(u_i - u_{i-1})}{(x_i - x_{i-1})} = u_{i-\frac{1}{2}}^m \left(\frac{u_i - u_{i-1}}{x_i - x_{i-1}}\right). \tag{5.5}$$

We choose this style of discretisation, since upon expansion the PME can be written in a hyperbolic form for which an upwind approximation is deemed suitable, namely

$$u_t = mu^{m-1}u_x^2 + u^m u_x.$$

Obviously our solution is not complete, since we have not yet stated how to evaluate the values of u at the current grid positions. The next section shows how the solution can be obtained from the current state of the mesh and a discrete integral of u relating to the original equidistribution idea.

5.1.2 The Porous Medium Solution

The moving mesh equation presented above (5.4 with 5.5) has already coupled together the prescription for the motion of the grid and the dynamics of the PME. Due to (5.2) the resulting system of ordinary differential equations should move the

grid in such a way that, as the material is diffused, computational cells hold an equal quantity of mass. Restating the equidistribution principle above, our grid should then, at all times, satisfy

$$\int_{x_i(t)}^{x_{i+1}(t)} u dx = \frac{1}{N} \int_{x_0(t)}^{x_N(t)} u dx = \theta \qquad i = 0, \ldots, N-1.$$

Using a trapezium rule approximation for the equidistributed mass $\theta(t)$ we have that a piecewise linear approximation will satisfy

$$\frac{1}{2}(u_{i+1} + u_i)(x_{i+1} - x_i) = \theta, \tag{5.6}$$

Since $u = 0$ at the foot of the moving boundary we can rearrange equation (5.6) to give us a sequence of algebraic equations yielding the approximate solution u in terms of the current grid co-ordinates and the constant mass θ,

$$\begin{aligned} u_N &= 0, \\ u_i &= \frac{2\theta}{(x_{i+1} - x_i)} - u_{i+1} \qquad i = N-1, \ldots, 0. \end{aligned} \tag{5.7}$$

Alternatively these algebraic relations can be written explicitly for u in summation form as

$$\begin{aligned} u_N &= 0, \\ u_i &= 2 \sum_{k=i+1}^{N} \theta \frac{(-1)^{k-i-1}}{x_k - x_{k-1}} \qquad i = N-1, \ldots, 0. \end{aligned}$$

Some compensation has to be made between the exact conserved equidistributed mass as used when deriving the system (5.4) and the discrete conserved mass θ used above. However if we start with a grid that has equidistributed discrete mass such that (5.6) hold over all cells, then appropriate discretisations of the ODE system will move the grid in such a way that this discrete approximation to the mass will be equidistributed and conserved. This is easily achieved by using a linearised form of the monitor function when generating the initial grid to complement the trapezium

rule approximation used in (5.6). We now proceed to deal with the moving boundary involved in the PME, in particular correctly approximating the speed of the moving front.

5.1.3 Speed of the Moving Boundary

To derive an approximation to the speed of the moving boundary, we consider conservation of mass over the entire region $[0, x_N]$. From the properties of the PME we have that

$$\frac{\partial}{\partial t}\int_0^{x_N(t)} u dx = 0$$

where $x_{N+1}(t)$ is the position of the front at time t. Differentiating, we have that

$$\int_0^{x_N(t)} u_t dx + u(x_N(t))\dot{x}_N = 0,$$

since the velocity of the node at $x = 0$ is zero (this is the centre of mass, which remains constant with time). Now substituting the PME into the integral on the left-hand side we have that

$$\int_0^{x_N(t)} (u^m u_x)_x dx + u(x_N(t))\dot{x}_N = 0$$

giving

$$u^m u_x|_{x=x_N} + u(x_N(t))\dot{x}_N = 0$$

since u_x is zero at $x = 0$ by symmetry. Rearrangement gives

$$\dot{x}_N = -(u^{m-1}u_x)|_{x=x_N}. \tag{5.8}$$

This formula can also be derived from the system of ODE's presented above in (5.4). Writing (5.4) as

$$u_i\dot{x}_i = u_{i-1}\dot{x}_{i-1} - [u^m u_x]_{x_i(t)}^{x_{i-1}(t)}$$

and adding the equations for $i = 0$ to N yields

$$\dot{x}_N = \frac{1}{u_N}\left(u_0\dot{x}_0 - [u^m u_x]_{x_0(t)}^{x_N(t)}\right),$$

which, after substituting in conditions regarding the stationary position and zero flux at x_0, gives equation (5.8). The velocity of the moving boundary is a result of total mass being preserved, this being a summation of the individual moving mesh equations. It follows then that our moving mesh system can be written as

$$\begin{aligned} \dot{x}_0 &= 0, \\ \dot{x}_i &= -(u^{m-1}u_x)|_{x=x_i} \qquad i = 1, \ldots, N. \end{aligned} \tag{5.9}$$

Finally, note that although $u = 0$ at $x = x_{N+1}$, the slope u_x is unbounded at this point, yielding a finite, non-zero front speed. Hence for the final term in the system (5.4), we have the discretisation

$$\dot{x}_N \approx u_{N-\frac{1}{2}}^{m-1}\frac{u_{N-1}}{x_N - x_{N-1}}. \tag{5.10}$$

5.1.4 Numerical Results II

The following section illustrates the capabilities of the moving mesh method when tackling the PME.

The system of ODE's (5.4) is solved using the NAG routine D02EJF [36], which uses a variable order, variable step size, backwards differentiation formula (BDF) to integrate the system forward in time. The routine simply requires a subroutine to be written in FORTRAN to provide the speeds of the nodes at any given time, the procedure for which is as follows. Given the value of the conserved, equidistributed mass $\theta(t)$ and the current gird co-ordinates $\underline{x}$

- Compute the current porous medium solution $\underline{u}$ using equation (5.7).
- Calculate the speed of the nodes using equation (5.4).

The routine provides its own internal adaptive time step and performs the integration to a user specified error tolerance. Our initial grid is computed using the

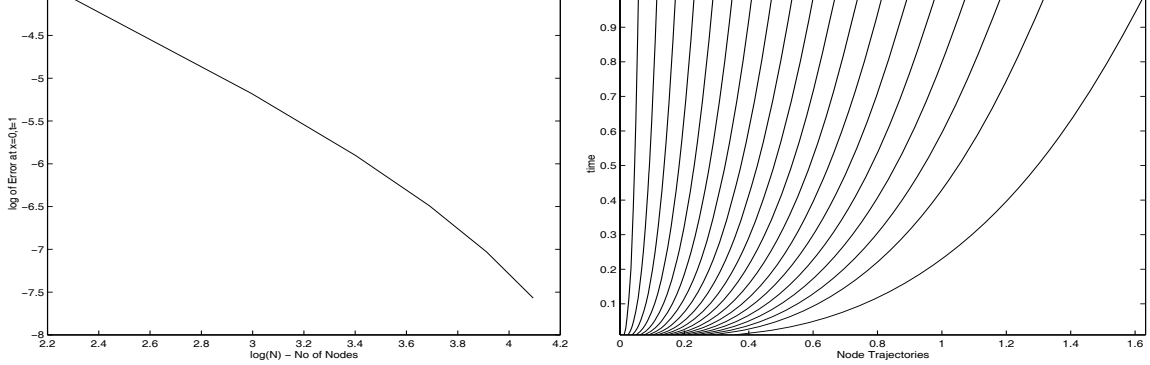

Figure 5.1: Error convergence and Node trajectories for $m = 1$

equidistribution algorithm as outlined by Baines [3], stated in Section 2.1, equation (2.6) with the aforementioned discrete version of the mass monitor given as

$$M_{i+\frac{1}{2}} = \frac{u(x_i, t) + u(x_{i+1}, t)}{2}$$

where the initial state is specified using the Murray solution (2.38). We compare our numerical solution against the analytical solution in a similar way as in Chapter 4. Using 20 nodes and setting $tstart = 0.01$, we compute the solution shown in Figure 5.2 for the gentle case when $m = 1$. The approximate solution values are denoted by the crossed curve whilst the Murray solution is represented by the solid line. By observation, one can see that the moving mesh method performs well. Most importantly the undesirable features from the regridding algorithm used previously have been taken care of. The moving boundary allows direct implementation of the Neumann conditions, and by construction the moving mesh points conserve mass. On the left hand side of Figure 5.1 we can see the convergence of the solution at $x = 0$ at $t = 1$ as N is increased. This measure of the error is thought to be of most significance. Since the solution u is recovered directly from the approximate value of the moving boundary and the algorithm moves backward towards this point, it follows that this point should carry the greatest truncation error. The graph is plotted as the logarithms of N and the error, and from the gradient of the curve we can see that the algorithm has second order accuracy. The right hand side of the figure shows the trajectories of the nodes.

Due to the simple power ($m = 1$) of the diffusion coefficient, the solution does not

form a steep front at the moving boundary. Hence nodes do not need to be placed directly near the moving boundary to improve accuracy in this region. Figure 5.3 shows the results when the power of the diffusion coefficient, m, is increased to 3. In this case the resolution of the grid near the boundary is too coarse, obviously having ill effects on the approximation of the wave speed and hence the resulting solution. Simply increasing the nodes will not deal effectively with the problem since due to the nature of the monitor function, nodes will be attracted to areas of high u and not to regions of great variation of u. The next section illustrates how this problem can be overcome with a slight adjustment to the moving mesh method.

Before we continue, we can show how the mesh produced for $m = 1$ reproduces the scaling invariance results covered in Section 2.3, Chapter 2. Budd suggests that by using the mass monitor the mesh will adhere to a scaling invariance property. Hence transforming the mesh co-ordinates and porous medium solution should produce an invariant mesh and solution with respect to time. Figure 5.4 shows the scaled mesh trajectories and the evolution of the scaled solution u. These figures are produced by transforming the numerical results displayed in Figure 5.2 to the invariance solution variables using equation (2.37) As can be seen from the graphs the computed mesh and discrete solution also exhibits this scaling invariance as expected.

5.2 Mass Conservation and Grid Refinement

The numerical solutions presented in Figure 5.3 illustrate that the method in its current state will not place nodes to accurately resolve the steep front formed at the moving boundary. However looking at the derivation of the method from a slightly different perspective gives us a tool to place moving nodes in this region. Equation (5.2) does not have to arise from an equidistribution principle. In fact we could choose any initial distribution of mass, i.e.

$$\int_{x_i(t)}^{x_{i+1}(t)} u dx = \theta_{i+\frac{1}{2}} \qquad i = 0, \ldots, N-1$$

where the masses $\theta_{i+\frac{1}{2}}$ could be any strategically chosen mass. The system of ODE's

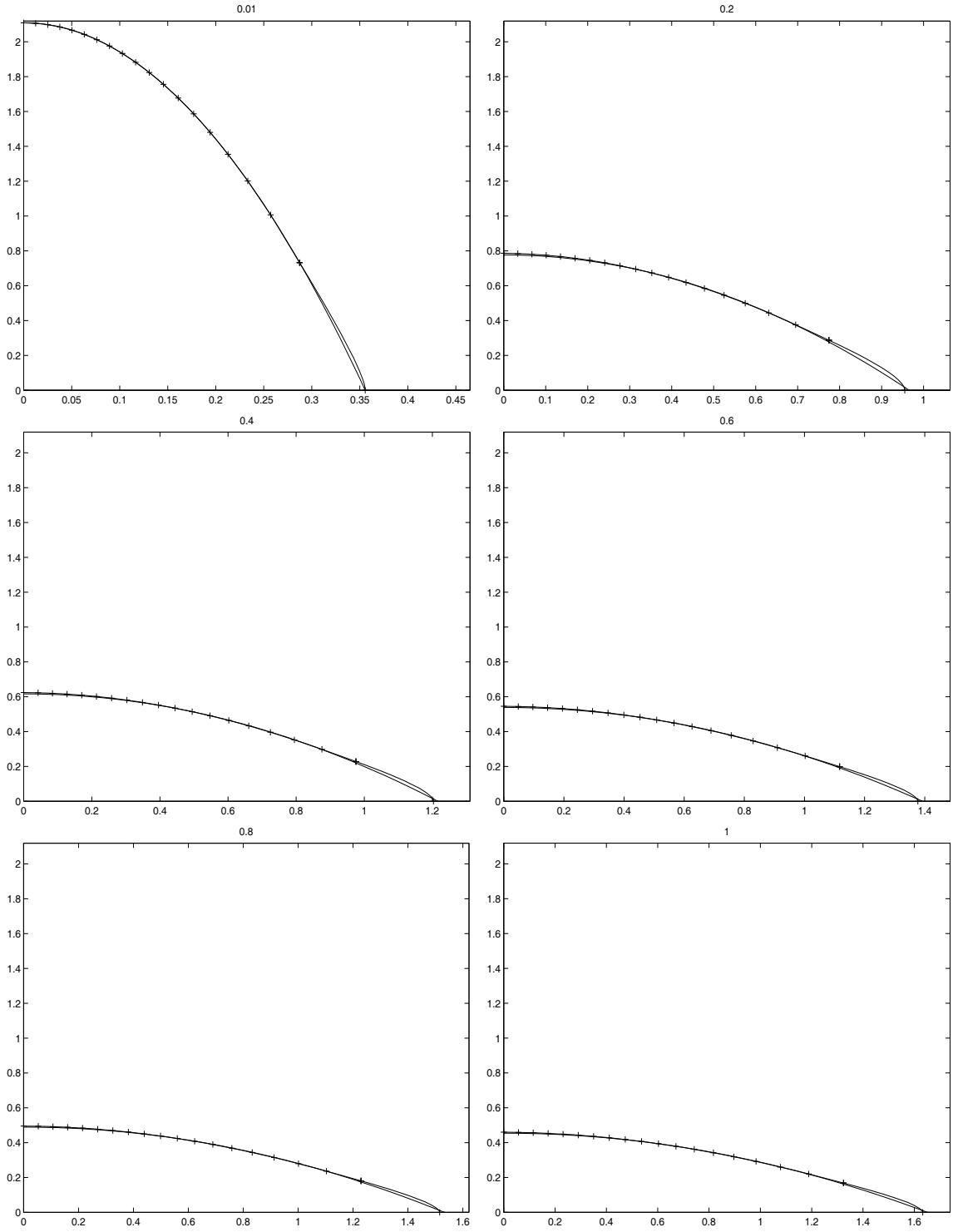

Figure 5.2: Approximate and reference PME solutions for $m = 1$

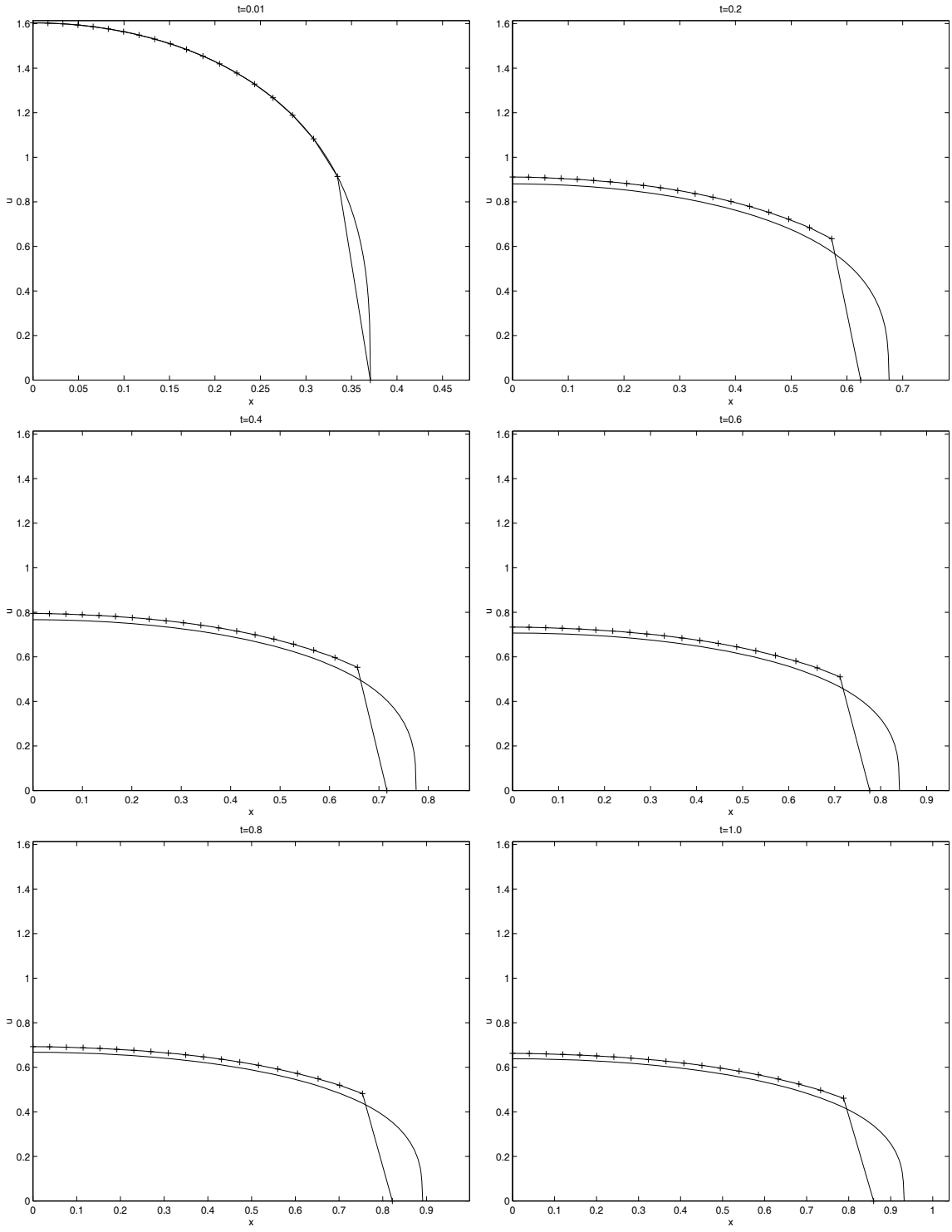

Figure 5.3: Approximate and reference PME solutions for $m = 3$

remains exactly the same, the only difference in the algorithm being a subtle change in the recovery of the u solution from the current mesh. We now have to store each mass $\theta_{i+\frac{1}{2}}$ and use the corresponding relations

$$\begin{aligned} u_N &= 0 \\ u_i &= \frac{2\theta_{i+\frac{1}{2}}}{(x_{i+1} - x_i)} - u_{i+1} \qquad i = N-1, \ldots, 0, \end{aligned} \tag{5.11}$$

which in turn can be written as

$$\begin{aligned} u_N &= 0 \\ u_i &= 2 \sum_{k=i+1}^{N} \theta_{i+\frac{1}{2}} \frac{(-1)^{k-i-1}}{x_k - x_{k-1}} \qquad i = N-1, \ldots, 0. \end{aligned}$$

In order to cluster nodes in the desired area, we simply choose to place smaller quantities of mass in these regions. In this case, we need to improve grid resolution near the moving boundary. Numerous techniques exist for choosing how to refine a stationary grid in this way (see Knupp & Steinberg [55]). However, for simplicity we choose to utilise parts of the FORTRAN code existing in the original procedure. We

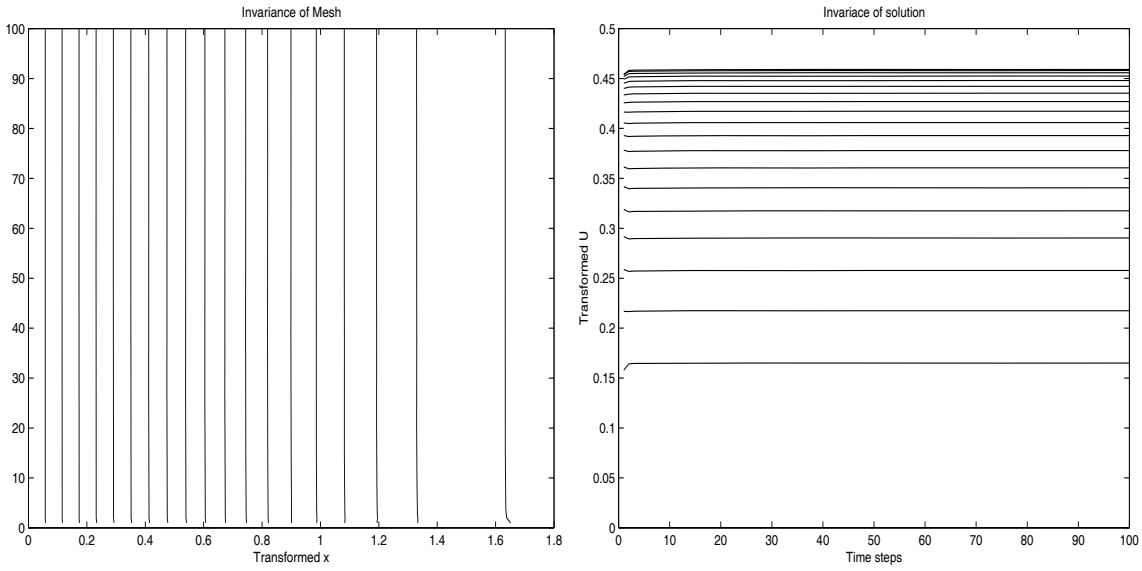

Figure 5.4: The invariant transformed computed mesh (Left) and PME solution (Right) for $m = 1$

refine the grid in the last computational cell, simply subdividing the grid by halving the mass contained in the cell repeatedly until the mass in the newly created final cell is below a specific tolerance *gridtol*. This is done by using the equidistribution ideas used in the previous section. We now present results using the refined quantities of mass.

5.2.1 Numerical Results III

We present sets of results for larger values of m ($m = 3$ and $m = 6$). Both demonstrate that the method can accurately resolve a steep front near the moving boundary. We begin where equidistributing the mass failed in section (5.1.4), with $m = 3$. Starting with an initial number of 15 nodes and adding an extra 10 by setting $gridtol = 10^{-4}$, we show in Figure 5.3 the numerical solution, again denoted by crosses, against the Murray solution shown with the solid line, for various times. Nodes are placed tightly in the intended region without affecting the existing qualities of the method. We push the method further by increasing the value of m to 6. In this case the front appears vertical with an area of high curvature at the peak of the wave. Again by using $gridtol = 10^{-4}$, this time starting with 10 nodes, an additional 9 points are added automatically. Again the method handles this more stringent test.

We can again illustrate the scaling invariance of the mass monitor using the appropriate transformations as before, despite equidistribution not being strictly adhered to over the entire mesh. Equidistribution is still enforced in all the unrefined cells, so we would expect under the transformations that the refinement nodes, mesh trajectories and solution would be bounded between those corresponding to the last and penultimate nodes on the unrefined mesh. Indeed Figure 5.5 shows exactly what we expect, the refinement nodes staying invariant too and their spacing seeming to reflect the diminishing quantities of mass held in each cell as they get closer to the boundary.

However, looking at the wider sphere of problems, it is not necessarily known where nodes should be added. Moreover it is possible that as the solutions develop in other problems, particular features of interest may move around the mesh, so

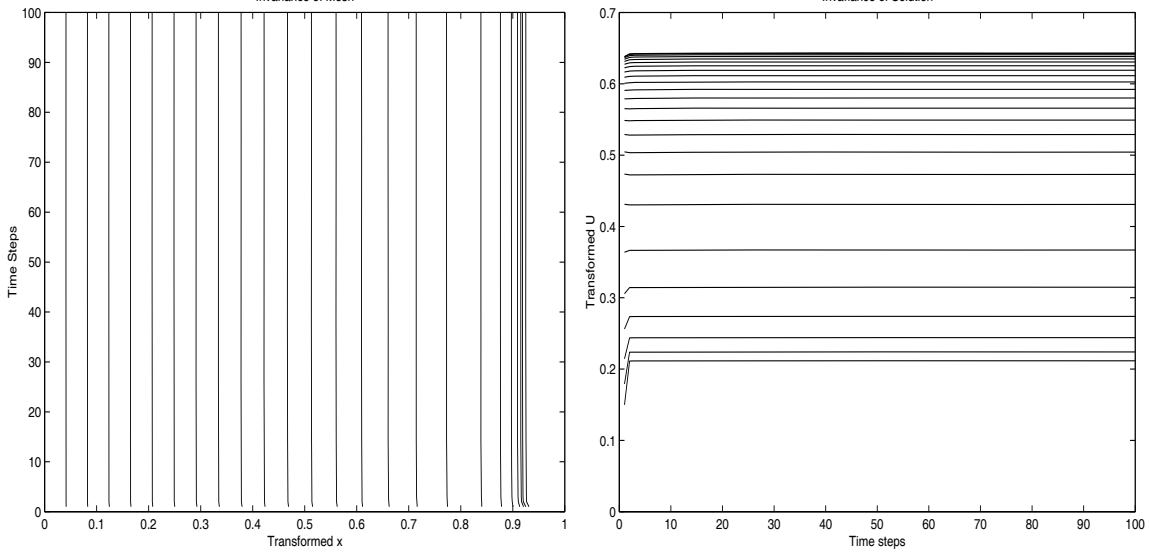

Figure 5.5: The invariant transformed computed mesh (Left) and porous medium solution (Right) for $m = 3$, using a refined grid.

requiring more nodes to be added in other cells. This could be overcome by modifying the method further by adding and removing nodes in different areas in time as required. This sort of adaptive routine has been well developed, especially in finite element methods. However this type of technique has repercussions in that continual changes needed to be made in the data structure of the computational code. It would be more convenient and efficient to be able to keep the number of nodes constant through computations whilst effectively dispersing nodes over the mesh. We now develop the ideas presented in this and the previous section by introducing a more effective monitor function with these aims in mind.

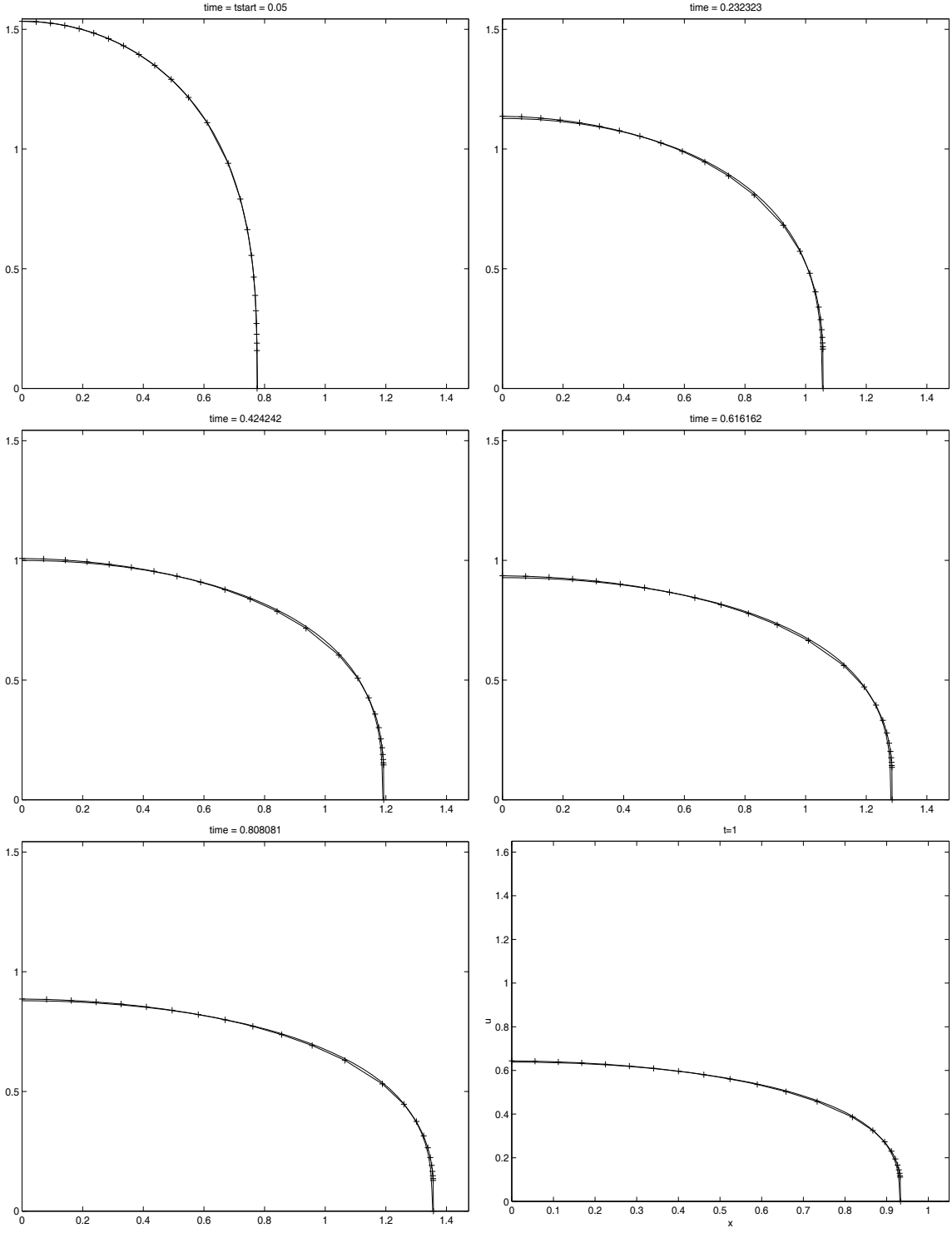

Figure 5.6: Approximate and reference PME solutions for $m = 3$, using grid refinement

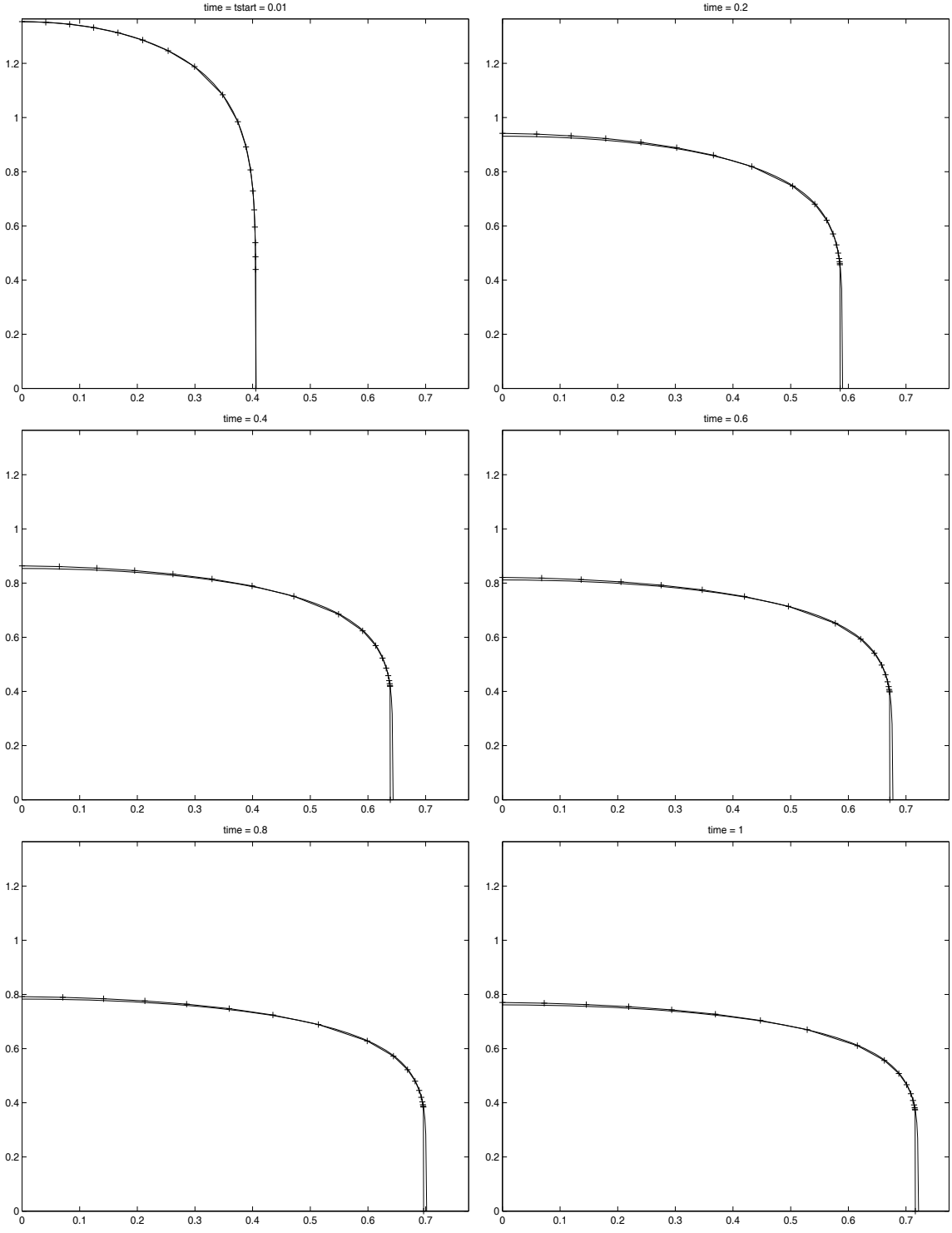

Figure 5.7: Approximate and reference PME solutions for $m = 6$ using grid refinement

5.3 Using the Gradient Monitor

We now extend the method to incorporate what we hope will be a more effective monitor function. The aim is to be able to accurately resolve features of high gradient without the need for grid refinement. In the previous sections the moving mesh method was first initiated, whilst noting that when a mass monitor function was used the potentially troublesome term θ_t becomes equal to zero. For other choices of monitor this is not necessarily the case. However, we are able to derive a method in much the same way as before, whilst maintaining the desired properties of the method, by taking advantage of geometric characteristics of the monitor function and the porous medium solution.

We propose a change in monitor to that used in the regridding process described in Chapter 3, namely,

$$M(u) = |u_x|.$$

Starting in the same manner as before, we state the new equidistribution principle,

$$\int_{x_i(t)}^{x_{i+1}(t)} u_x dx = \frac{1}{N}\int_{x_0(t)}^{x_N(t)} u_x dx = \theta(t) \qquad i = 0, \ldots, N-1. \tag{5.12}$$

Since our porous medium solution will be monotonic we can drop the modulus signs from the monitor function, and upon differentiation with respect to time, we have,

$$\int_{x_i(t)}^{x_{i+1}(t)} u_{xt} dx + \dot{x}_{i+1}(t) u_x|_{x_{i+1}(t)} - \dot{x}_i(t) u_x|_{x_i(t)} = \theta_t \tag{5.13}$$

The right hand term can be dealt with by noticing the distributive properties of the gradient monitor. As described in previous chapters, nodes will be placed equally over the solution range of u. Since our solution will be monotonic, we can write from (5.12)

$$\theta(t) = \frac{1}{N}(u(x_0) - u(x_N))$$

giving, via substitution of the PME and using $u_{x_N} = 0$,

$$\begin{aligned}\theta_t &= \frac{1}{N}\frac{\partial u}{\partial t}|_{x_0} \\ &= \frac{1}{N}(u^m u_x)_x|_{x_0}. \end{aligned} \tag{5.14}$$

Returning to equation (5.13), we derive a moving mesh equation in a similar fashion as used for the previous monitor function, giving

$$[(u^m u_x)_x]_{x_i(t)}^{x_{i+1}(t)} + \dot{x}_{i+1}(t)u_x|_{x_{i+1}(t)} - \dot{x}_i(t)u_x|_{x_i(t)} = \frac{1}{(N-1)}(u^m u_x)_x|_{x=0}.$$

This leads to a corresponding ODE system to solve,

$$\begin{aligned}\dot{x}_0 &= 0 \\ \dot{x}_i u_x|_{x_i} &= \frac{1}{N}(u^m u_x)_x|_{x=0} \\ &+ \dot{x}_{i-1}u_x|_{x_i-1} - [(u^m u_x)_x]_{x_i}^{x_i-1} \qquad i = 1, \ldots, N \end{aligned} \tag{5.15}$$

In a similar way as presented in the section 5.1.3, the expression for the speed of the moving boundary will be valid when using the above expression. It could also be derived by considering the time differentiation of (5.12) over the entire domain. However for a more simpler expression we elect to continue using the expression derived for use with the mass monitor (5.10). The next section illustrates why this expression is valid by the using the mass conservation property to relate the current solution u to the grid x and the quantity $\theta(t)$.

5.3.1 The Equidistributed Quantity

When using the mass monitor (without refinement), the quantity $\theta(t)$ was simply the equidistributed conserved quantity of mass in each cell. With the change of monitor comes a change in this measure $\theta(t)$. We have already derived an expression for the time derivative of this integral term. We now develop an understanding of its role in coupling the current grid and the porous medium solution.

From the known geometric properties of an equidistributed grid with respect to the gradient monitor and taking into consideration the porous medium solution, we can write

$$\begin{aligned}\int_{x_i(t)}^{x_{i+1}(t)} u_x dx &= \theta(t) \\ &= \frac{1}{N}(u_0 - u_N) \\ &= \frac{u_0}{N} \qquad i = 0, \ldots N-1.\end{aligned}$$

Written another way, we have that

$$u_i(t) = (N-i)\theta(t) \qquad for \qquad i = 0, \ldots N. \tag{5.16}$$

We now consider the composite trapezium rule expression for the total conserved mass, TM say, i.e.

$$TM = \frac{1}{2}\sum_{j=0}^{N-1}(u_{j+1}(t) + u_j(t))(x_{j+1} - x_j). \tag{5.17}$$

Substituting in equation (5.16) gives

$$TM = \frac{\theta(t)}{2}\sum_{j=0}^{N-1}(2(N-j)-1)(x_{j+1} - x_j)$$

which when rearranged provides us with an expression for the value of $\theta(t)$

$$\theta(t) = \frac{TM}{\Phi} \tag{5.18}$$

where

$$\Phi = \left(\sum_{j=0}^{N-1}(x_{j+1} - x_j)(N-j)\right) - \frac{1}{2}(x_N - x_1).$$

Given the grid x at any time, we can compute $\theta(t)$ using (5.18) and then recover the appropriate solution values u using (5.16). This procedure is used in the NAG routine for solving the system of ODE's (5.15) along with the following discretisations

$$u_x|_{x_i} = \frac{1}{2}\left[\left(\frac{u_{i+1} - u_i}{x_{i+1} - x_i}\right) + \left(\frac{u_i - u_{i-1}}{x_i - x_{i-1}}\right)\right], \tag{5.19}$$

$$(u^m u_x)_x|_{x_i} = \frac{2}{(x_{i+1} - x_{i-1})}\left[u^m_{i+\frac{1}{2}}\left(\frac{u_{i+1} - u_i}{x_{i+1} - x_i}\right) - u^m_{i-\frac{1}{2}}\left(\frac{u_i - u_{i-1}}{x_i - x_{i-1}}\right)\right]. \tag{5.20}$$

5.3.2 Numerical Results IV

We can use the gradient monitor to resolve the steep moving boundary without the need for grid refinement. Figure 5.9 presents the results in the same manner as before. From these results we can clearly see how the nodes arrange themselves equally over the solution of range of u as expected. The left hand side of Figure 5.8 illustrates how the nodes are moved towards the developing front as desired. However when compared to results using the mass monitor with or without refinement, there are clearly inaccuracies between the generated and analytic solution.

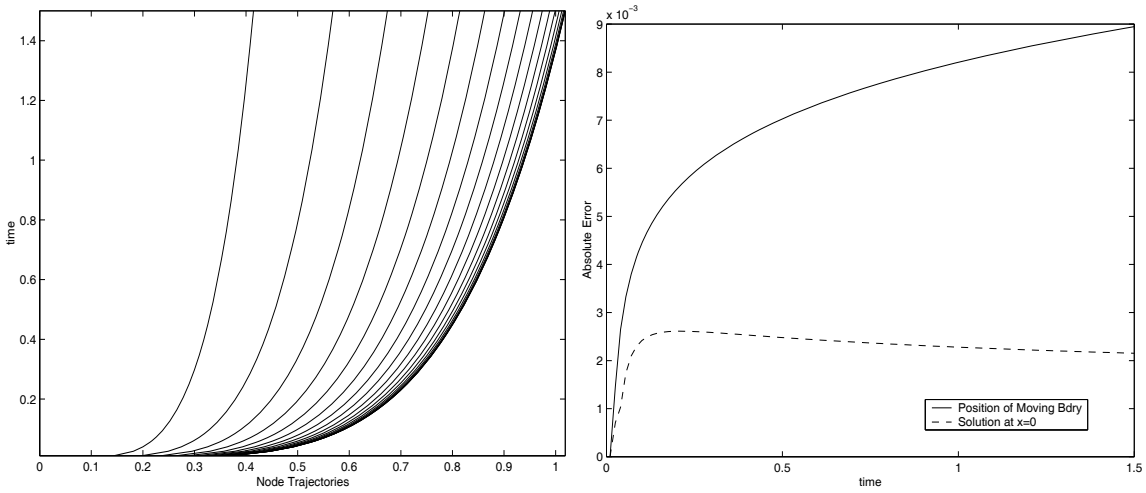

Figure 5.8: Node trajectories (left) and error measures (right) for the gradient monitor, $m = 3$.

The right hand side of Figure 5.8 shows two error measures from the results shown in Figure 5.9. The development over time of the absolute error in both the position of moving boundary and the solution at $x = 0$ are shown. Both show a steep rise in the early stages of the computation. We can investigate how the approximations to these quantities may be linked via the monitor. We used the simplest estimate to the velocity of x_N, i.e.

$$\dot{x}_N \approx u_{N-\frac{1}{2}}^{m-1} \frac{u_{N-1}}{x_N - x_{N-1}}$$

which, using expression (5.16) can be written as

$$\begin{aligned} \dot{x_N} &= \left(\frac{\theta(t)}{2}\right)^{(m-1)} \left(\frac{\theta(t)}{(x_N - x_{N-1})}\right), \\ &= \frac{\theta(t)^m}{2^{(m-1)}(x_N - x_{N-1})}. \end{aligned}$$

In this way, the position of the boundary is directly connected to the value of the quantity $\theta(t)$. Further study of Figures 5.9 and 5.8 show how the moving mesh method attracts nodes onto the vertical section of the front, leaving areas of coarse resolution near the boundary. Moreover, our approximation to the time derivative of $\theta(t)$ will involve a spatial discretisation in this region. It becomes apparent then that poor local approximations to this term have had severe repercussions on the global solution. These apparent poor approximations near the origin have the worst effect early in the computations where the diffusion is fast, which would explain the rapid growth in error illustrated in Figure 5.8. As the solution progresses and changes, the variation of u at $x = 0$ becomes gentle and, the error at this point decreases, although the error in the front position increases.

It is clear then that, for a truly accurate solution to the PME for values of $m > 1$, the grid needs to maintain a reasonable resolution globally whilst forcing $(x_j - x_{j-1})$ to be small locally at the steep front. This is exactly what the grid refinement process achieved when used in conjunction with the mass monitor.

In the final section of this chapter we introduce a composite of the two previous monitors in an attempt to dampen the effects of the gradient monitor and provide a more suitable moving mesh for these more demanding problems.

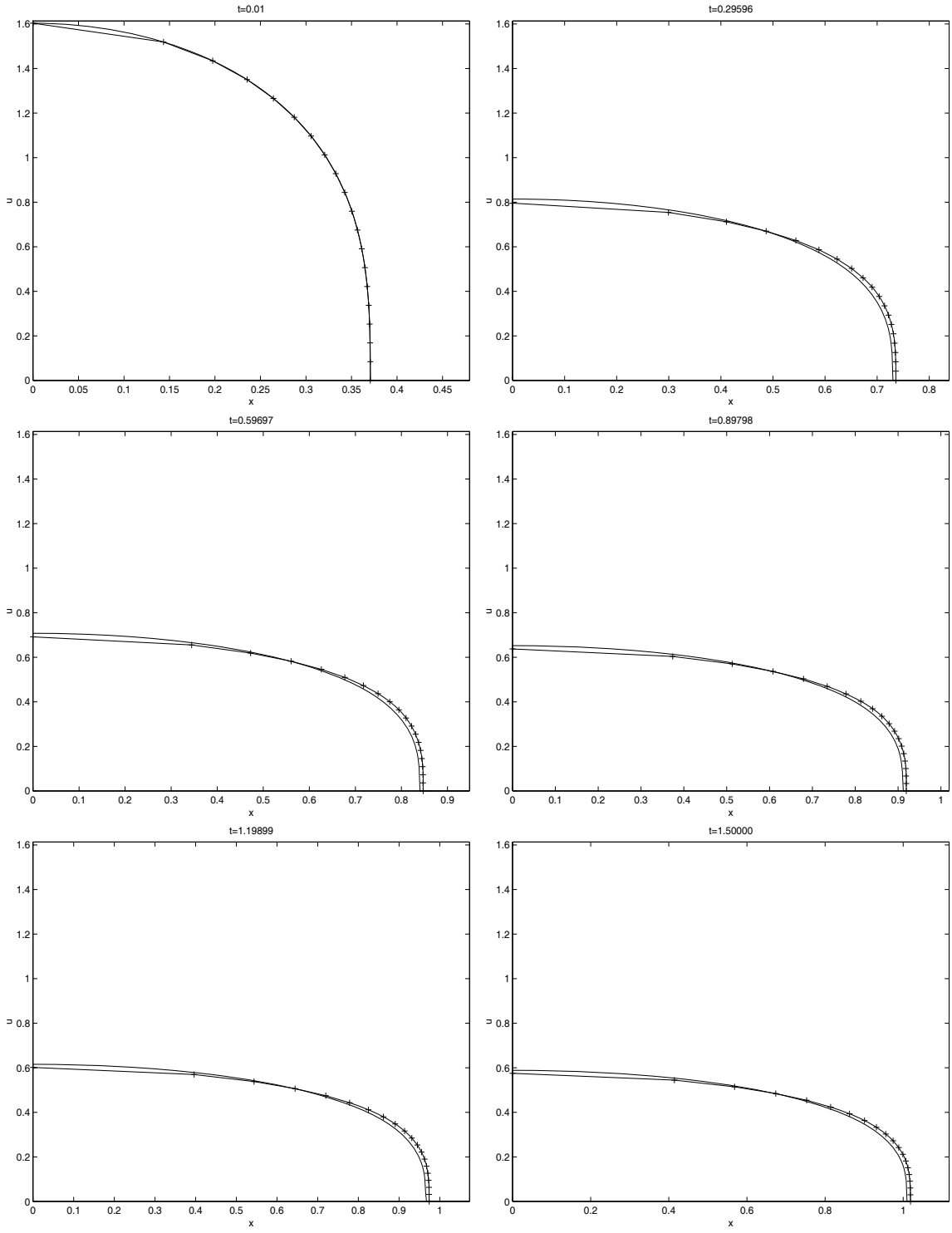

Figure 5.9: Approximate and reference PME solutions using gradient monitor, $m = 3$

5.4 A Combination Monitor

Findings from the previous section suggest that in order for our moving mesh method to provide good approximations to the PME over the whole domain, we must be able to accurately resolve the moving steep front whilst retaining a reasonable spatial resolution globally for the equidistributed quantity $\theta(t)$. We now attempt to combine the mass and gradient monitors to achieve this aim.

It seems sensible to suggest this methodology. We have seen that the gradient monitor will cluster points in region of large variations in u, whilst the mass monitor will attract points to section of the grid where the discrete integral of u is large, namely the origin. Hence we propose the combination monitor function,

$$M(u) = u + |u_x|.$$

Since the magnitude of the derivative become very large near the boundary and is negative everywhere, we rewrite the monitor, with the aid of a weighting parameter ω, in the more effective form

$$M(u) = u - \omega u_x. \tag{5.21}$$

The derivation of the moving mesh equations follows in the same way as before. In fact, as we shall see later, the equations will be linear combinations of those presented in (5.4) and (5.15). First we turn our attention to the θ_t term. Once again, starting with the equidistribution rule, again using the properties of the porous medium solution, we have

$$\begin{aligned}
\theta(t) &= \int_{x_i(t)}^{x_{i+1}(t)} (u - \omega u_x)\, dx \qquad i = 1, \ldots N-1 \\
&= \frac{1}{N} \int_0^{x_N(t)} (u - \omega u_x)\, dx \\
&= \frac{1}{N} \left(\int_0^{x_N} u dx - \omega \int_0^{x_N} u_x dx \right) \\
&= \frac{1}{N} \left(\int_0^{x_N} u dx + \omega u_0 \right).
\end{aligned}$$

Hence upon differentiation with respect to time, owing to mass conservation, the integral of the right hand side disappears, leaving us with

$$\begin{aligned}\theta_t &= \frac{\omega}{N}\frac{\partial u}{\partial t}|_{x=0}, \\ &= \frac{\omega}{N}\left(u^m u_x\right)_x|_{x=0}.\end{aligned}$$

We can now construct our moving mesh equation, differentiating the equidistribution principle to obtain

$$\frac{\partial}{\partial t}\int_{x_i}^{x_{i+1}}(u-\omega u_x)dx = \theta_t.$$

Differentiating the integral and substituting in (5.22) yields

$$\int_{x_i(t)}^{x_{i+1}(t)}\left(u_t-\omega u_{xt}\right)dx + \left[\dot{x}\left(u-\omega u_x\right)\right]_{x_i(t)}^{x_{i+1}(t)} = \frac{\omega}{N}\left(u^m u_x\right)_x|_{x=0}.$$

Hence, substituting the PME gives

$$\begin{aligned}\frac{\omega}{N}(u^m u_x)_x|_{x=0} &= \left[(u^m u_x)-\omega(u^m u_x)_x\right]_{x_i(t)}^{x_{i+1}(t)} \\ &+ \dot{x}_{i+1}\left(u_{i+1}-\omega u_x|_{x_{i+1}}\right) \\ &+ \dot{x}_i\left(u_i-\omega u_x|_{x_i}\right).\end{aligned}$$

Our ODE system now becomes

$$\begin{aligned}\dot{x}_0 &= 0, \\ \dot{x}_i\left(u_i-\omega u_x|_{x_i}\right) &= \frac{\omega}{N}(u^m u_x)_x|_{x=0} \\ &- \left[(u^m u_x)-\omega(u^m u_x)_x\right]_{x_i-1}^{x_i} \\ &+ \dot{x}_{i-1}\left(u_{i-1}-\omega u_x|_{x_i-1}\right).\end{aligned} \tag{5.22}$$

To complete the method we need to construct a relation between the grid x and $\theta(t)$. By comparison with the method outlined in Section 5.3, we now have the modified discrete form of the equidistribution principle over each cell $i = 0, \ldots, N-1$,

$$\theta(t) = \int_{x_i}^{x_{i+1}}(u-\omega u_x)dx \approx \frac{1}{2}(u_{i+1}+ui)(x_{i+1}-x_i) - \omega(u_{i+1}-u_i), \tag{5.23}$$

which when rearranged gives

$$u_i = \frac{\theta(t)}{\gamma_{i+1}} + u_{i+1}\frac{\mu_{i+1}}{\gamma_{i+1}} \tag{5.24}$$

where we introduce notation

$$\gamma_{i+1} = \omega + \frac{1}{2}(x_{i+1} - x_i), \tag{5.25}$$

$$\mu_{i+1} = \omega - \frac{1}{2}(x_{i+1} - x_i). \tag{5.26}$$

Back-substituting u_{i+1} and then u_{i+2} into (5.24) gives

$$\begin{aligned}
u_i &= \frac{\theta(t)}{\gamma_{i+1}} + \frac{\mu_{i+1}}{\gamma_{i+1}}\left[\frac{\theta(t)}{\gamma_{i+2}} + u_{i+2}\frac{\mu_{i+2}}{\gamma_{i+2}}\right], \\
&= \frac{\theta(t)}{\gamma_{i+1}} + \frac{\mu_{i+1}\theta(t)}{\gamma_{i+1}\gamma_{i+2}} + \frac{\mu_{i+1}\mu_{i+2}}{\gamma_{i+1}\gamma_{i+2}}u_{i+2}, \\
&= \frac{\theta(t)}{\gamma_{i+1}} + \frac{\mu_{i+1}\theta(t)}{\gamma_{i+1}\gamma_{i+2}} + \frac{\mu_{i+1}\mu_{i+2}}{\gamma_{i+1}\gamma_{i+2}}\left[\frac{\theta(t)}{\gamma_{i+3}} + u_{i+3}\frac{\mu_{i+3}}{\gamma_{i+3}}\right], \\
&= \frac{\theta(t)}{\gamma_{i+1}} + \frac{\mu_{i+1}\theta(t)}{\gamma_{i+1}\gamma_{i+2}} + \frac{\mu_{i+1}\mu_{i+2}\theta(t)}{\gamma_{i+1}\gamma_{i+2}\gamma_{i+3}} + \frac{\mu_{i+1}\mu_{i+2}\mu_{i+3}}{\gamma_{i+1}\gamma_{i+2}\gamma_{i+3}}u_{i+2}.
\end{aligned}$$

Further back substitution then gives us

$$u_i = \frac{\theta(t)}{\gamma_{i+1}}\left[1 + \prod_{k=i+1}^{i+1}\frac{\mu_k}{\gamma_{k+1}} + \prod_{k=i+1}^{i+2}\frac{\mu_k}{\gamma_{k+1}} + \ldots + \prod_{k=i+1}^{N-2}\frac{\mu_k}{\gamma_{k+1}}\right] + u_{N-1}\prod_{k=i+1}^{N-1}\frac{\mu_k}{\gamma_k}. \tag{5.27}$$

Since $u_N = 0$, so that $u_{N-1} = \frac{\theta(t)}{\gamma_N}$, substituting this final relation into the equation above we have a general expressions for u_i,

$$\begin{aligned}
u_N &= 0 \\
u_{N-1} &= \frac{\theta(t)}{\gamma_N} \\
u_i &= \frac{\theta(t)}{\gamma_{i+1}}\left[1 + \prod_{k=i+1}^{i+1}\frac{\mu_k}{\gamma_{k+1}} + \prod_{k=i+1}^{i+2}\frac{\mu_k}{\gamma_{k+1}} + \ldots + \prod_{k=i+1}^{N-1}\frac{\mu_k}{\gamma_{k+1}}\right] \\
&= \frac{\theta(t)}{\gamma_{i+1}}\left[1 + \sum_{j=i+1}^{N-1}\prod_{k=i+1}^{j}\frac{\mu_k}{\gamma_{k+1}}\right].
\end{aligned} \tag{5.28}$$

In a similar way as when formulating the method for the gradient monitor, we again consider the discrete total mass TM and substitute the expressions for u_i. Since $\theta(t)$ is a common factor in the expression for the general u_i, we can obtain the following expression giving a relation between $\theta(t)$ and the grid x,

$$\theta(t) = \frac{2TM}{\Psi} \tag{5.29}$$

where

$$\begin{aligned}\Psi \quad = \quad & \sum_{i=1}^{N-3}(x_{i+1} - xi)\left[\frac{1}{\gamma_{i+2}} + \frac{1}{\gamma_{i+1}} + \frac{1}{\gamma_{i+2}}\sum_{j=i+2}^{N-1}\prod_{k=i+2}^{j}\frac{\mu_k}{\gamma_{k+1}} + \frac{1}{\gamma_{i+1}}\sum_{j=i+1}^{N-1}\prod_{k=i+1}^{j}\frac{\mu_k}{\gamma_{k+1}}\right] \\ & + (x_{N-1} - x_{N-2})\left[\frac{1}{\gamma_N} + \frac{1}{\gamma_{N-1}}\left(1 + \frac{\mu_{N-1}}{\gamma_N}\right)\right] \\ & + \frac{1}{\gamma_N}(x_N - x_{N-1}).\end{aligned}$$

As previously, when using the mass monitor, in order for our formulation to be valid initially, we need to have an initial grid which will initially satisfy the discrete relation (5.23). Our monitor will therefore take the form

$$M_{i+\frac{1}{2}} = \frac{u_i + u_{i+1}}{2} - \omega\frac{u_{i+1} - ui}{x_{i+1} - x_i}.$$

5.4.1 Numerical Results V

We are now in a position to review the effectiveness of combining the two previous forms of monitor. It is hoped that by utilising the properties of both monitors we shall be able to overcome the shortcomings of the gradient monitor whilst being able to resolve the steep fronts without the need for grid refinement. Figure 5.10 shows that at first glance this has been achieved. Generating the approximate solution in the usual manner, we present a solution for the PME when $m = 3$, choosing the parameter $\omega = 0.2$. Since the monitor and hence the resulting moving mesh equations are linear combinations of the two previous systems, it makes sense to use a linear combination of the discretisations used to solve the two systems respectively. Hence the solutions are discretised using the previous expressions (5.5),(5.19) and (5.20). We can see clearly that the combination monitor has performed well for this

value of m. Both the position of the front and the global solution are seemingly well approximated. This success carries on further when considering the more challenging problem when $m = 5$. Here ω is chosen to have a value of 0.1. The resulting numerical solutions are presented in Figure 5.11.

In Section 5.3, we speculated that the gradient monitor, whilst clustering nodes directly at the moving steep front, did not provide adequate mesh resolution near the origin. In that case, the method permitted the velocity of the moving boundary to be written in terms of $\theta(t)$, the time derivative of which is approximated at $x = 0$. Hence the resolution of the mesh in this region could have a global effect on the position of the moving boundary and the resulting solution. This relationship will also exist to some extent in the method for the combination monitor. It was hoped that by combining the two monitors a reasonable mesh spacing near the origin could be retained. Since consistent discretisations are used in both methods, we can compare the accuracy in the two methods in approximating θ_t. Figure 5.12 shows the relative error in approximating this term when generating the solutions shown in Figures 5.10 and 5.9. We use the relative error to give a better comparison since the parameter ω will reduce the magnitude of θ_t when applying the combination monitor. The graph shows that the amalgamated monitor function provides a better approximation to this term, the right-hand side of the figure suggesting that the finer resolution from the resulting combination mesh plays a major role in this improvement. Furthermore Figure 5.13 shows the node trajectories and the same error measures as illustrated in the Section 5.3.2 for the gradient monitor. The trajectories confirms that globally the mesh retains a sensible resolution, whilst the partial involvement of the gradient monitor provides an adequate grid spacing near the moving boundary in order to resolve the steep front. The second graph, when compared with the corresponding plot in Figure 5.8, shows that this combination results in significantly more accurate results in terms of the position of the propagating boundary and the solution of u at $x = 0$.

The left hand side of Figure 5.14 shows the variations of the absolute error in u at $x = 0$ as the value of ω is increased for three values of m in the porous media diffusion co-efficient. We are interested in which value of ω is in some sense 'best'.

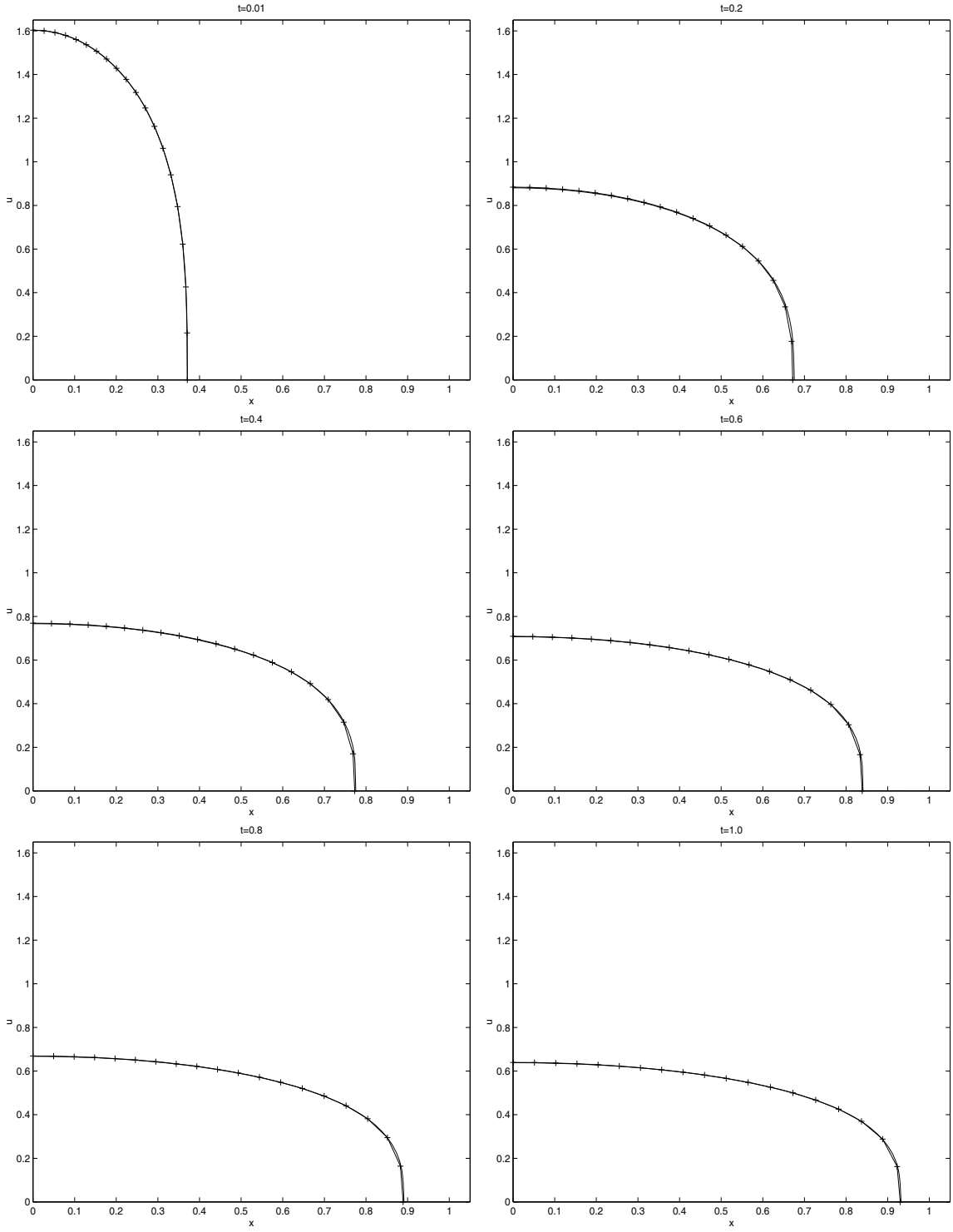

Figure 5.10: Approximate and reference PME solutions using the combination monitor, $m = 3$ and $\omega = 0.2$.

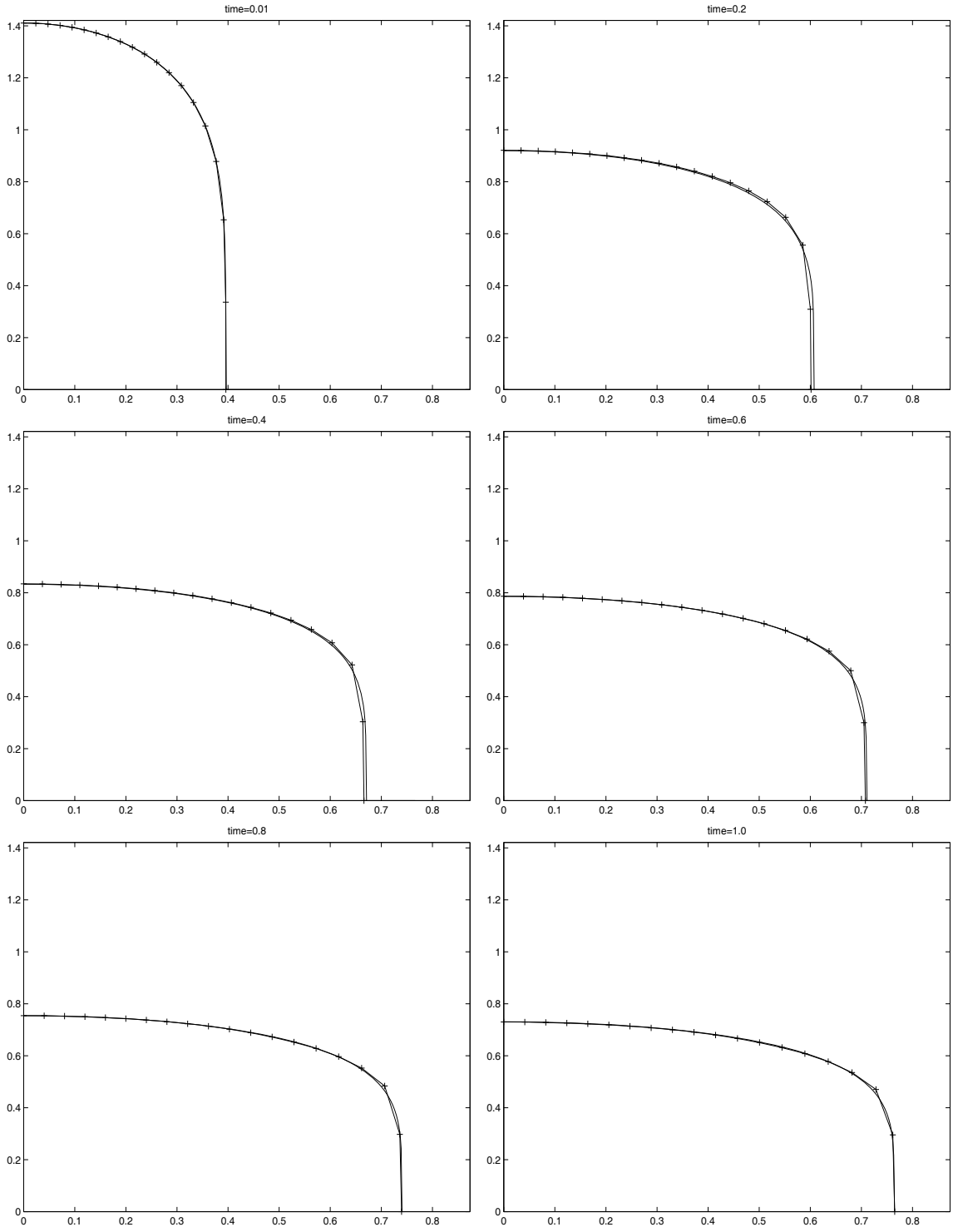

Figure 5.11: Approximate and reference PME solutions using the combination monitor, $m = 5$ and $\omega = 0.1$.

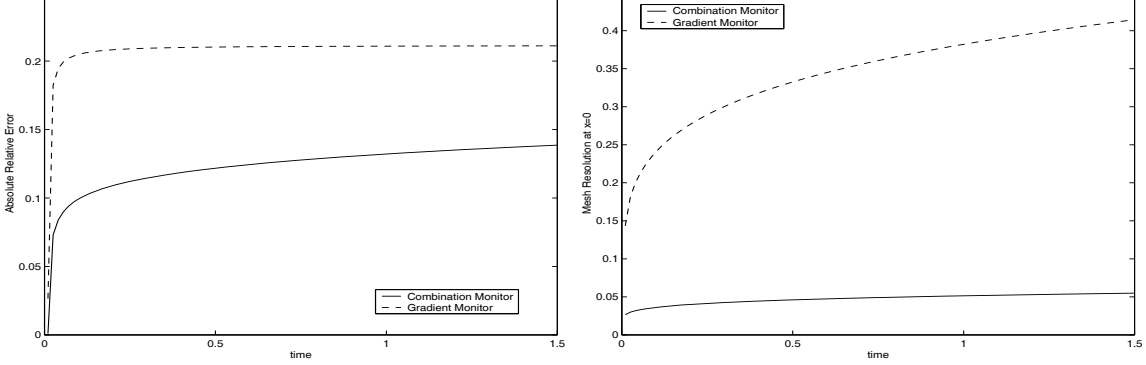

Figure 5.12: Absolute relative error in approximating θ_t using the gradient and combination monitor

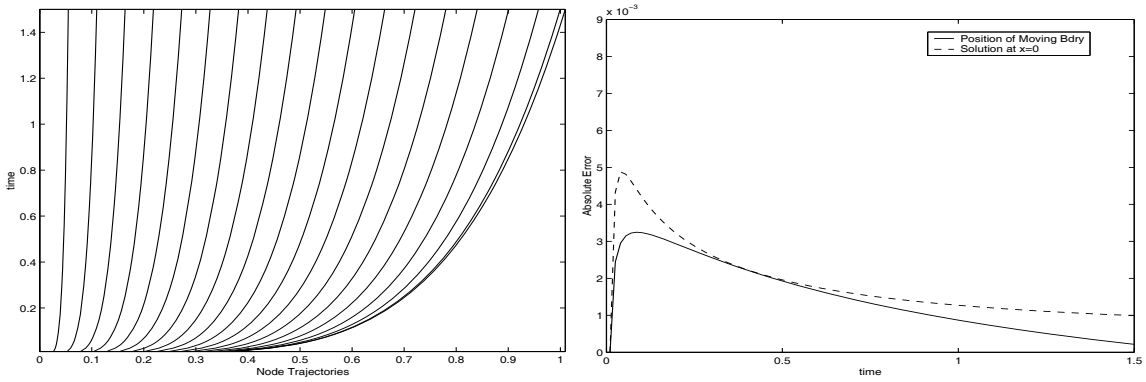

Figure 5.13: Node trajectories (left) and error measures (right) for the combination monitor, $m = 3$ and $\omega = 0.2$.

Looking at the minimum values of the three curves plotted, the graph suggests that as m increases from one to three, the 'optimum' ω increases and then soon decreases as m is raised further. The apparent initial increase in the most appropriate value of the parameter seems sensible since between these values the severity of the steep front increases dramatically. However the profile of the front continues to steepen with m past this point and yet our results suggest a smaller value of ω should be chosen. One possible explanation is the numerical error when approximating the integrand terms in our system (5.22). As the front steepens to become almost vertical (typical of large m), then these terms become very large indeed and are therefore extremely difficult to approximate. Hence choosing a smaller ω is likely to dampen any numerical errors incurred here. The right-hand graph in Figure 5.14 shows a polynomial best fit curve for this 'optimal' choice of ω depending on the

value of m. Fifty separate values of ω were choice in equal increment for over 40 equally spaced values of m between 0.1 and 4.2. The best fit curve has order 19 and was generated using MATLAB.

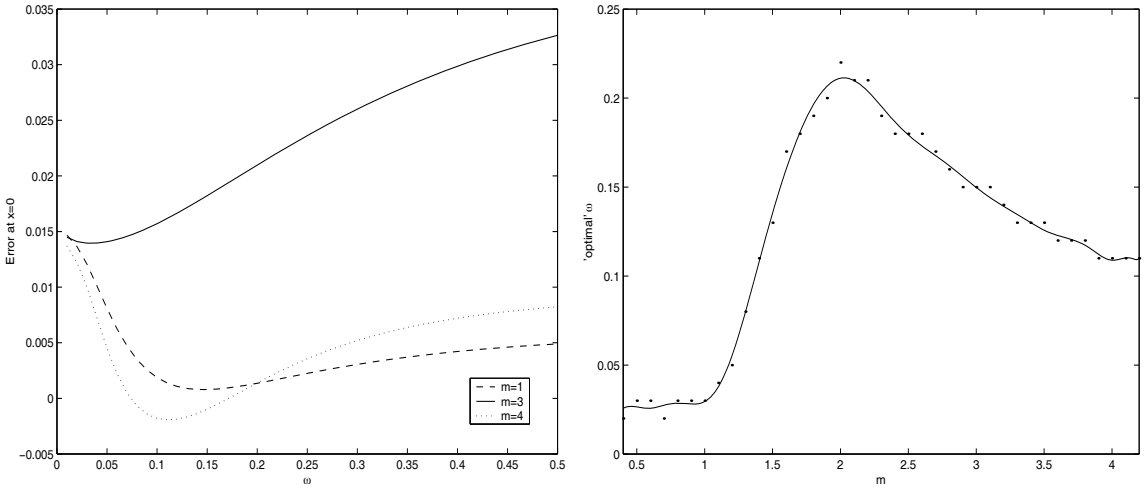

Figure 5.14: Error Measure with choice of ω for values of m.(Right). Polynomial best fit of 'optimal' ω for m. (Left)

5.5 Summary

We now summarise the work presented in this chapter, in particular highlighting the development of the moving mesh method with regard to the solution of the PME.

In the previous chapter, we were concerned with the solution of both the PME and the semiconductor problem. Also included in this chapter were numerical solutions to these applications generated via the contour zoning approach. It was concluded that in these problems which conserve mass, this method was quite unsuitable since errors introduced through interpolation when moving nodes resulted in the numerical mass varying with time. Moreover, when considering the PME, the algorithm did not easily let us deal with the moving boundary involved. Hence we sought a procedure which would move the mesh whilst exactly conserving mass and allow for a moving boundary.

Initially a moving mesh approach was used by equidistributing the mass monitor $M(u) = u$. This monitor has been used in the literature since it accords with the

inherent scaling properties of the PME. However, it was also noticed that by using this function, the moving mesh equations could be derived without the need for a traditional underlying computational mesh. The resulting method involves the forward integration in time of a system of ODE's governing the speeds of the current mesh coordinates. From the resulting grid, the solution u could be recovered using the conserved equidistributed mass θ. By equidistributing mass, good solutions were produced for the gentle porous media problem when $m = 1$. However, for larger values of m the mesh would not provide a fine enough grid resolution near the moving boundary to resolve the formation of the steep front. Hence we modified the method by dropping the equidistribution constraint on the mesh and moving the grid by conserving mass in each cell. By initially generating a grid with fine scale resolution in the appropriate regions the front was resolved with great success whilst retaining the global accuracy of previous methods.

In Section 5.3 we introduced the gradient monitor $M(u) = u_x$. It was hoped that using this monitor function would eliminate the need for grid refinement in the method. The moving mesh method in this case was derived in a similar manner as for $M(u) = u$, taking advantage of the properties of the porous medium solution and, more importantly, keeping the mass conserved. The method now involved the evaluation of the now time dependent equidistributed quantity $\theta(t)$, a discrete approximation to the 'measure' of the monitor contained in each cell. Despite the attractive features of this monitor, it was found that the gradient monitor clustered nodes too tightly near the front, leaving regions near the origin sparse. Due to the way in which the solution u was related to the grid and θ_t, this had repercussions on the approximation of the velocity of the moving boundary and the resulting approximate solution.

Finally we managed to combine the two monitors and obtained a happy balance between the two properties of each function. Using a parameter ω to scale the influence of the gradient monitor, the resulting algorithm produced accurate approximation for both small and large values of m. Numerical results and accuracy measures were presented and indeed drove all stages of the development. In particular, in Section 5.4.1 we attempted to explain the 'optimum' choice of ω.

It is apparent that at present the method is highly specific to the solution of the PME. However the method does manage to tackle all the deficiencies of the previous regridding solution. In the following chapter we apply the method to the semiconductor problem and introduce an application involving a source term with the aim of replicating previous numerical and theoretical approximations to the time of solution blow up.

Chapter 6

The Moving Mesh Method for Further Applications

In the previous chapter we followed the development of a moving grid method specifically for the solution of the PME. Whilst the principles of the mesh movement are viable when tackling other problems, the existence of a known solution value, e.g. $u = 0$, at the moving boundary in the PME allowed us to recover the global solution directly from the calculated grid via an algebraic equation. The method also took advantage heavily of the mass conservation properties of the equation. For these reasons the method could be seen to be too problem specific and lacking in robustness.

The present chapter illustrates how the method can be extended to solve problems without these inherent properties. The first problem considered is the semiconductor model problem we are already familiar with from Chapter 4. In this case, the equation does conserve mass but has a Neumann boundary condition imposed at the fixed boundaries, hence we do not know a value of u at any of these points from which to construct our resulting solution. We shall nevertheless construct the moving grid method as before, later further modifying the monitor function in order to more accurately replicate the initial conditions and including a local solution process for the boundary value of u from which we can build the dopant profile.

In the second half of this chapter we introduce a problem with a non-linear source term which involves a solution 'blowing up' at some finite time T. This behaviour

is caused by the source term and hence mass is no longer conserved. Here we shall adapt the original method which equidistributed mass (see Section 5.1), but couple a derivative term for the total mass with the system of ODE's prescribing the mesh movement. We shall evaluate the performance of the method by comparing blow-up times with previous numerical estimates from the literature.

6.1 The Semi-Conductor Problem Revisited

It was concluded in Chapter 4 that for a numerical method to adequately solve the semiconductor problem the method itself must conserve mass exactly. With this in mind the dynamic grid method introduced in the previous chapter seems to suit this application perfectly. However, the solution will now have a time dependent solution value at both boundaries. When considering the PME the zero value of u at the moving boundary permitted an obvious computational saving since the solution could be derived directly from the grid and the quantity $\theta(t)$ via an algebraic relation (see Sections 5.1.2, 5.3.1 and 5.4), without a separate time integration for the solution u. Ideally we would like to retain this characteristic of the existing algorithm.

To begin the derivation of the moving grid method for this problem we will first state our solution strategy with particular reference to the solution in regions of low concentrations of dopant. We then move on to the derivation of the moving mesh equation, introducing a further adaption to the monitor presented in Section 5.4 of the previous chapter. Finally we present our numerical results with reference to a fine scale stationary mesh solution.

6.1.1 Solution Technique

We begin by recalling the model semiconductor problem. Defined on the fixed region $x \in [0, 1]$, the diffusion of a dopant through silicon is modelled by the equation

$$u_t = ((u + \epsilon)u_x)_x, \tag{6.1}$$

ϵ being a constant of the order 10^{-2}. Neumann conditions are imposed at each boundary, $x = 0$ and $x = 1$, and the dopant has initial Gaussian distribution

$$u(x, 0) = e^{-\sigma x^2}.$$

Previously we have chosen σ to have a value of 50 and we continue to do so.

The combination of a fixed domain and Neumann boundary conditions force u to fluctuate away from a value of zero at $x = 1$. In order for our method to allow us to trace back from this point and form the solution from the grid, we need to have a handle on this value u at this point. Due to the low concentration and gradient at this region of interest, we propose to make use of the resulting low temporal changes in u at this boundary and implement a local explicit finite-difference solution. We begin by deriving an expression for the time derivative of u at this point

We shall again be using a grid consisting of $N + 1$ nodes numbered $x_0, \ldots, x_N$ where $x_0 = 0$ and $x_N = 1$. With reference to Section 3.2.1, Chapter 3, we consider the integral form of equation (6.1) in the localised region of the control volume of node x_N. As before this region is formally defined as the length between x_N and the cell midpoint $x_{N-\frac{1}{2}}$.

Given the Neumann conditions imposed at the right hand boundary we have that

$$\begin{aligned} \int_{x_{N-\frac{1}{2}}}^{x_N} u_t dx &= \int_{x_{N-\frac{1}{2}}}^{x_N} \left((u+\epsilon)u_x\right)_x dx \\ &= -(u+\epsilon)u_x|_{x_{N-\frac{1}{2}}} \end{aligned}$$

Discretising in an upwind manner as before, we derive an approximate expression for u_t via

$$\begin{aligned} (x_N - x_{N-\frac{1}{2}})u_t &= -(u^n_{N-\frac{1}{2}} + \epsilon)\left[\frac{u_N - u_{N-1}}{x_N - x_{N-1}}\right], \\ u_t &= -2(u^n_{N-\frac{1}{2}} + \epsilon)\left[\frac{u_N - u_{N-1}}{(x_N - x_{N-1})^2}\right]. \end{aligned} \tag{6.2}$$

Despite the low gradient of the solution at $x = 1$ a reasonable mesh spacing will always be required near that point during the solution to ensure accuracy of the value u_N and hence the global solution. Taking this into consideration, we propose a further modification to the combination monitor used in the previous chapter. We now need to ensure that the mesh is adequately represented over the regions of low concentration that initially lies away from the Gaussian maximum value. If we consider the combination monitor used previously

$$M(u) = u - \omega u_x$$

it is obvious that the value of the monitor will diminish quickly in regions of low concentration. Moreover, due to the nature of the initial conditions, nodes will be clustered directly inside the Gaussian. If we were to use such a monitor, the initial distribution would not cover the areas of interest involved in the solution of u_N and furthermore the foot of the Gaussian is unlikely to be well represented. The second issue raised here is crucial in the evolution of the numerical approximation since, as mentioned before, of particular interest in this problem is the amount of material diffused in regions well away from the main body of dopant. An accurate solution must be able to simulate the correct amount of material 'leaking' from the bottom section of the Gaussian (the parameter ϵ in the model equation (6.1) controls this leakage). Hence it is vital that the solution be well represented in this region too, preferably without the step-like ramp representations found in the regridding solution in Chapter 3. Moreover good representation in these areas will also improve the accuracy of the conserved mass. With these reasons in mind, we propose to add an extra constant term α to the monitor function which will help to control the equal spread of the nodes,

$$M(u) = u - \omega u_x + \alpha. \tag{6.3}$$

The inclusion of this extra term will ensure that nodes will be located away from the Gaussian centre, as is required for a good solution of u_N. Moreover it will help to place nodes nearer the foot of the initial profile. Figure 6.1 illustrates the effect of introducing the quantity α into the monitor function. Irrespective of the choice

of ω, the inclusion of α puts points in regions where the other terms in the previous monitor would have a negligible value. Although these distributions are simple examples using only 15 nodes and setting $\sigma = 50$, they effectively demonstrate the intended purpose of the additional term. As a result the conserved discrete mass should be a more accurate estimate. We continue by deriving the new algebraic relations between the moving grid x, the solution u and the quantity $\theta(t)$ associated with the modified combination monitor.

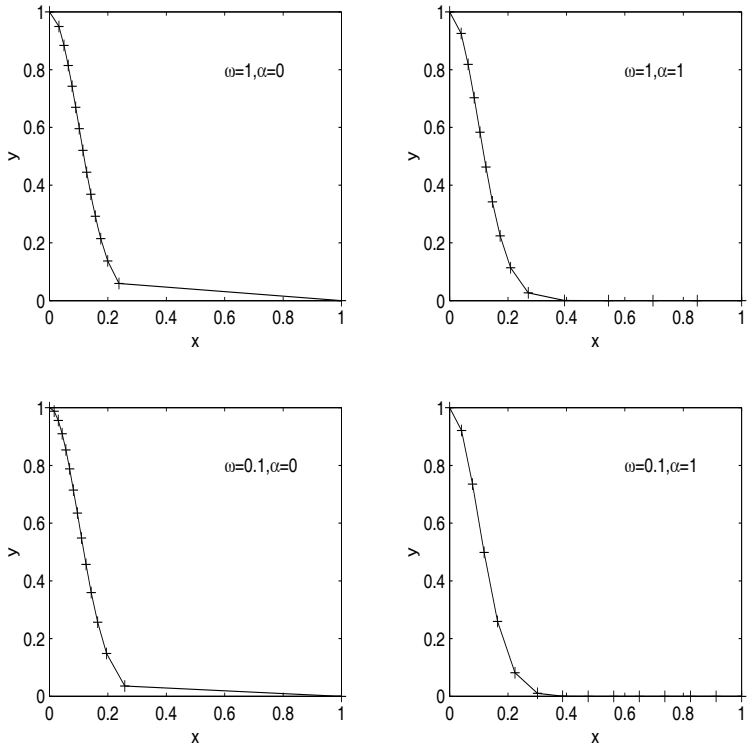

Figure 6.1: Initial distributions with varying choices of ω and α for the modified monitor (6.3).

As before we express $\theta(t)$ as a discrete approximation to the integral of the monitor over all cells $i = 0, \ldots, N-1$, i.e.

$$\begin{aligned} \theta(t) &= \int_{x_i}^{x_{i+1}} (u - \omega u_x + \alpha) dx \\ &\approx \frac{1}{2}(u_{i+1} + ui)(x_{i+1} - x_i) - \omega(u_{i+1} - u_i) + \alpha(x_{i+1} - x_i). \end{aligned} \tag{6.4}$$

Reiterating the notation used in Chapter (5.4) and introducing the extra term, we have

$$\begin{aligned}
\gamma_{i+1} &= \omega + \frac{1}{2}(x_{i+1} - x_i), \\
\mu_{i+1} &= \omega - \frac{1}{2}(x_{i+1} - x_i), \\
\delta_{i+1} &= \alpha(x_{i+1} - x_i).
\end{aligned}$$

and, upon rearrangement equation (6.4) gives us

$$u_i = \frac{\theta(t)}{\gamma_{i+1}} + u_{i+1}\frac{\mu_{i+1}}{\gamma_{i+1}} - \frac{\delta_{i+1}}{\gamma_{i+1}}. \tag{6.5}$$

Using forward substitution into this system we have

$$\begin{aligned}
u_i &= \frac{\theta(t)}{\gamma_{i+1}} - \frac{\delta_{i+1}}{\gamma_{i+1}} + \frac{\mu_{i+1}}{\gamma_{i+1}}\left[\frac{\theta(t)}{\gamma_{i+2}} - \frac{\delta_{i+2}}{\gamma_{i+2}} + u_{i+2}\frac{\mu_{i+2}}{\gamma_{i+2}}\right], \\
&= \frac{\theta(t)}{\gamma_{i+1}}\left[1 + \frac{\mu_{i+1}}{\gamma_{i+2}}\right] - \frac{1}{\gamma_{i+1}}\left[\delta_{i+1} + \delta_{i+2}\frac{\mu_{i+1}}{\gamma_{i+2}}\right] + \frac{\mu_{i+1}\mu_{i+2}}{\gamma_{i+1}\gamma_{i+2}}u_{i+2}.
\end{aligned}$$

It is obvious that upon further substitution we will have a similar expression to that derived for the original combination monitor. So assuming we have given the newly calculated value for u_N given via integrating equation (6.2), we have a general expression for the solution, i.e.

$$\begin{aligned}
u_i &= \frac{\theta(t)}{\gamma_{i+1}}\left[1 + \prod_{k=i+1}^{i+1}\frac{\mu_k}{\gamma_{k+1}} + \prod_{k=i+1}^{i+2}\frac{\mu_k}{\gamma_{k+1}} + \ldots + \prod_{k=i+1}^{N-1}\frac{\mu_k}{\gamma_{k+1}}\right] - \\
&\quad \frac{1}{\gamma_{i+1}}\left[\delta_{i+1} + \delta_{i+2}\prod_{k=i+1}^{i+1}\frac{\mu_k}{\gamma_{k+1}} + \delta_{i+3}\prod_{k=i+1}^{i+2}\frac{\mu_k}{\gamma_{k+1}} + \ldots + \delta_N\prod_{k=i+1}^{N-1}\frac{\mu_k}{\gamma_{k+1}}\right] \\
&\quad + u_N\prod_{k=i+1}^{N}\frac{\mu_k}{\gamma_k}, \\
&= \frac{\theta(t)}{\gamma_{i+1}}\left[1 + \sum_{j=i+1}^{N-1}\prod_{k=i+1}^{j}\frac{\mu_k}{\gamma_{k+1}}\right] \\
&\quad + \frac{1}{\gamma_{i+1}}\left[\delta_{i+1} - \sum_{j=i+1}^{N}\delta_j\prod_{k=i+1}^{j}\frac{\mu_k}{\gamma_{k+1}}\right] + u_N\prod_{k=i+1}^{N}\frac{\mu_k}{\gamma_k}.
\end{aligned}$$

Hence upon substituting this into the trapezium rule for the total conserved mass TM (5.17), we have that the required expression for $\theta(t)$ is

$$\theta(t) = \frac{2TM - u_N\Psi + \Omega}{\Phi} \tag{6.6}$$

where

$$\begin{aligned}\Phi &= \sum_{i=0}^{N-2}(x_{i+1}-x_i)\left[\frac{1}{\gamma_{i+1}}\left(1+\sum_{j=i+1}^{N-1}\prod_{k=i+1}^{j}\frac{\mu_k}{\gamma_{k+1}}\right)+\frac{1}{\gamma_{i+2}}\left(1+\sum_{j=i+2}^{N-1}\prod_{k=i+2}^{j}\frac{\mu_k}{\gamma_{k+1}}\right)\right] \\ &\quad + \frac{(x_N - x_{N-1})}{\gamma_N},\end{aligned}$$

$$\begin{aligned}\Omega &= \sum_{i=0}^{N-2}(x_{i+1}-x_i)\left[\frac{1}{\gamma_{i+1}}\left(\delta_{i+1}+\sum_{j=i+1}^{N-1}\delta_j\prod_{k=i+1}^{j}\frac{\mu_k}{\gamma_{k+1}}\right)\right] \\ &\quad + \sum_{i=1}^{N-2}(x_{i+1}-x_i)\left[\frac{1}{\gamma_{i+2}}\left(\delta_{i+2}+\sum_{j=i+2}^{N-1}\delta_j\prod_{k=i+1}^{j}\frac{\mu_k}{\gamma_{k+1}}\right)\right] \\ &\quad + (x_N - x_{N-1})\frac{\delta_N}{\gamma_N},\end{aligned}$$

and

$$\Psi = \sum_{i=0}^{N-2}(x_{i+1}-x_i)\left[\prod_{k=i+1}^{N}\frac{\mu_k}{\gamma_k}+\prod_{k=i+2}^{N}\frac{\mu_k}{\gamma_k}\right] + (x_N - x_{N-1})\left[1+\frac{\mu_N}{\gamma_N}\right].$$

We now have the required relation between the grid, the equidistributed quantity $\theta(t)$ and the solution u. Although the expressions above seem rather complicated to compute, they are still mere algebraic relations which are easier to compute compared to the full integration in time of the underlying PDE coupled with the moving mesh system. We now continue further with the derivation of the moving mesh system for the semi-conductor equation with the modified combination monitor.

6.1.2 The Moving Mesh

Since the semi-conductor problem is defined on a fixed domain, we will have different boundary constraints on our resulting ODE system,

$$\dot{x_0} = \dot{x_N} = 0.$$

In the same manner as before, we start with the equidistribution principle over all cells,

$$\theta(t) = \int_{x_i(t)}^{x_{i+1}(t)} (u - \omega u_x + \alpha)\, dx \qquad i = 0, \ldots, N-1.$$

Differentiating with respect to t and expanding the right hand side gives

$$\frac{d\theta}{dt} = \int_{x_i(t)}^{x_{i+1}(t)} (u - \omega u_x)_t\, dx + \left[(u - \omega u_x + \alpha)\, \dot{x}\right]_{x_i}^{x_{i+1}}$$

Upon using the PDE (6.1) terms and re-arranging, we obtain

$$\begin{aligned} (u - \omega u_x + \alpha)|_{x_{i+1}} \dot{x}_{i+1} &= \frac{d\theta}{dt} + \dot{x}_i (u - \omega u_x + \alpha)|_{x_i} \\ &\quad - \left[(u+\epsilon)u_x - \omega((u+\epsilon)u_x)_x\right]_{x_i}^{x_{i+1}} \; i = 0, \ldots, N-1. \end{aligned} \tag{6.7}$$

Considering the time derivative of $\theta(t)$, we look at the integral of the monitor over the entire domain,

$$\begin{aligned} (N-1)\frac{d\theta}{dt} &= \int_0^1 (u - \omega u_x + \alpha) dx \\ &= \frac{d}{dt}\int_0^1 u dx - \omega \frac{d}{dt}\int_0^1 u_x dx + \alpha \frac{d}{dt}\int_0^1 dx. \end{aligned}$$

Since the boundaries are fixed, the time differentiation carries through to inside the integrals without extra terms. Furthermore, removing the first and last integrals (relating to conservation of mass, and the fixed domain respectively), we are left to evaluate the remaining term

$$(N-1)\frac{d\theta}{dt} = -\omega \frac{d}{dt}(u_N - u_1).$$

Finally we find that upon substituting in the original PDE 6.1) we have

$$\frac{d\theta}{dt} = -\frac{\omega}{(N-1)}\left[((u+\epsilon)u_x)_x\right]_{x=0}^{x=1}, \tag{6.8}$$

noting that due to the conditions imposed at the boundary $x = 1$, this formula now involves a term from the right hand side. However this contribution is strongly linked to the evaluation of the new solution value u_N at this point (6.2). We now have all the necessary components with which to complete the construction of the ODE system for the motion of the mesh. So putting everything together and substituting (6.8) into (6.7) we have,

$$\begin{aligned}
\dot{x}_1 &= 0, \\
(u - \omega u_x + \alpha)|_{x_{i+1}} \dot{x}_{i+1} &= -\frac{\omega}{(N-1)} \left[((u+\epsilon)u_x)_x\right]_{x=0}^{x=1} + \dot{x}_i (u - \omega u_x + \alpha)|_{x_i} \\
&\quad - \left[(u+\epsilon)u_x - \omega((u+\epsilon)u_x)_x\right]_{x_i}^{x_{i+1}} \\
&\quad i = 1, \ldots, N-2, \qquad (6.9) \\
\dot{x}_N &= 0.
\end{aligned}$$

As with the previous systems, we start with an equidistributed mesh relating to a discrete version of the monitor function. In this case the appropriate grid is defined using the approximate monitor

$$M_{i+\frac{1}{2}} = \frac{u_i + u_{i+1}}{2} - \omega \frac{u_{i+1} - ui}{x_{i+1} - x_i} + \alpha(x_{i+1} - x_i),$$

using the equidistribution algorithm by outlined in equation (2.6) in Chapter 2.1. Since the same terms are involved as in the equations for the combination monitor we shall use the same discretisations stated in the previous chapter, (5.5), (5.19) and (5.20).

Our solution however is not complete without the integration of equation (6.2). In order to maintain a level of consistency in our time integration of this value in conjunction with the mesh movement, we simply add the equation (6.2) as an extra component in the ODE system. This leaves us with the system of variables

$$(\dot{x}_0, \dot{x}_1, \dot{x}_2, \ldots, \dot{x}_{N-1}, \dot{x}_N, u_t).$$

The procedure works in exactly the same way as that for finding the node positions. So in providing the NAG routine with the correct derivative values we undertake the following at each time step.

- Start with the current grid $(x_0, \ldots, x_N)$, a discrete approximation to the mass TM, and the solution value u_N.

- Using (6.6), compute $\theta(t)$ and then in turn calculate the current global solution $u_0, \ldots, u_N$ from (6.5).

- Compute the speeds of the nodes using (6.9).

- Find the time derivative of the solution at $x = 1$ using (6.2).

6.1.3 Numerical Results VI

We shall analyse the performance of the moving mesh method in this application by computing an approximate solution on a stationary regular mesh consisting of 100001 nodes. The solution is generated using a semi-implicit scheme with an adaptive time-stepping approach, as outlined for the solution of this problem in Chapter 3. We have computed these stationary mesh solutions for three different values, $\epsilon = 0.5, 0.01, 0.005$, for comparison with the moving mesh solutions. Before we look at the resulting solutions and the effect of adjusting the leakage of material in the concentration profile, we turn our attention to the error in the semiconductor solution with reference to our fine scale computation.

The top graphs in Figure 6.2 show the absolute error measures at both boundaries. We would expect, given a certain level of accuracy in the solution at $x = 1$, that a further truncation error would be incurred, since the global solution was reconstructed from the computed points using the algebraic relation (6.5). However from these plots the magnitude of error incurred at both boundaries looks to be of the same order. Both convergence charts suggest that the method is again of second order accuracy. The value of ϵ does seem to have an effect, with large errors incurred for the faster rate of leakage, even though all cases were integrated forward in time using a tolerance of 10^{-16} in the NAG routine.

Figure 6.3 shows the generated solutions for $\epsilon = 0.01$. The initial mesh is generated and then moved by setting the other parameter in the monitor function to be $\omega = \alpha = 1$. In the full semiconductor problem several initial profiles of dopant are created. Although the solutions were generated over the domain $[0, 1]$, we plot the

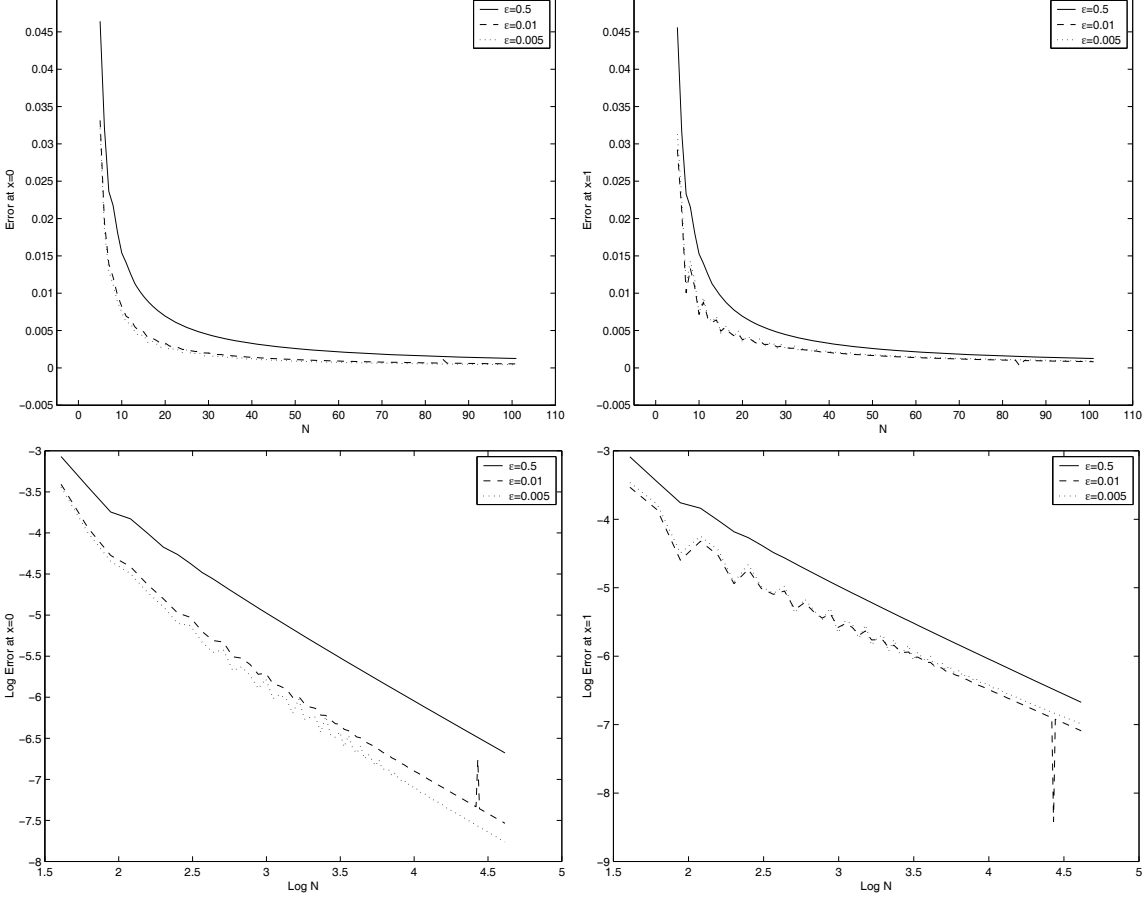

Figure 6.2: Absolute error at $x = 1$ and $x = 0$ (Top). Logarithmic plots of the error to show convergence (Bottom)

profile with its reflection in $x = 1$ to illustrate the node movement when two significant quantities of material reach the far boundary, as happens in the real problem. In Figure 6.4 we plot the trajectories for the three values of ϵ. As expected the larger the value of ϵ the quicker the diffusion acts, since material leaks from the main body of the Gaussian at a faster rate. The top plot shows how for $\epsilon = 0.5$ the nodes move very fast initially and the profile develops into what appears to be a steady state solution. The modified combination monitor drives the mesh to be equally spaced as this state of the solution develops. As the value of ϵ is decreased, the rate of this development slows down. The bottom plot for $\epsilon = 0.005$ shows how the nodes move towards the centre as a significant amount of dopant reaches $x = 1$. As the value of u_N gets larger and the state approaches, these nodes spring back as the mesh springs back to becoming equally spaced, i.e. $M(u) \to 1$.

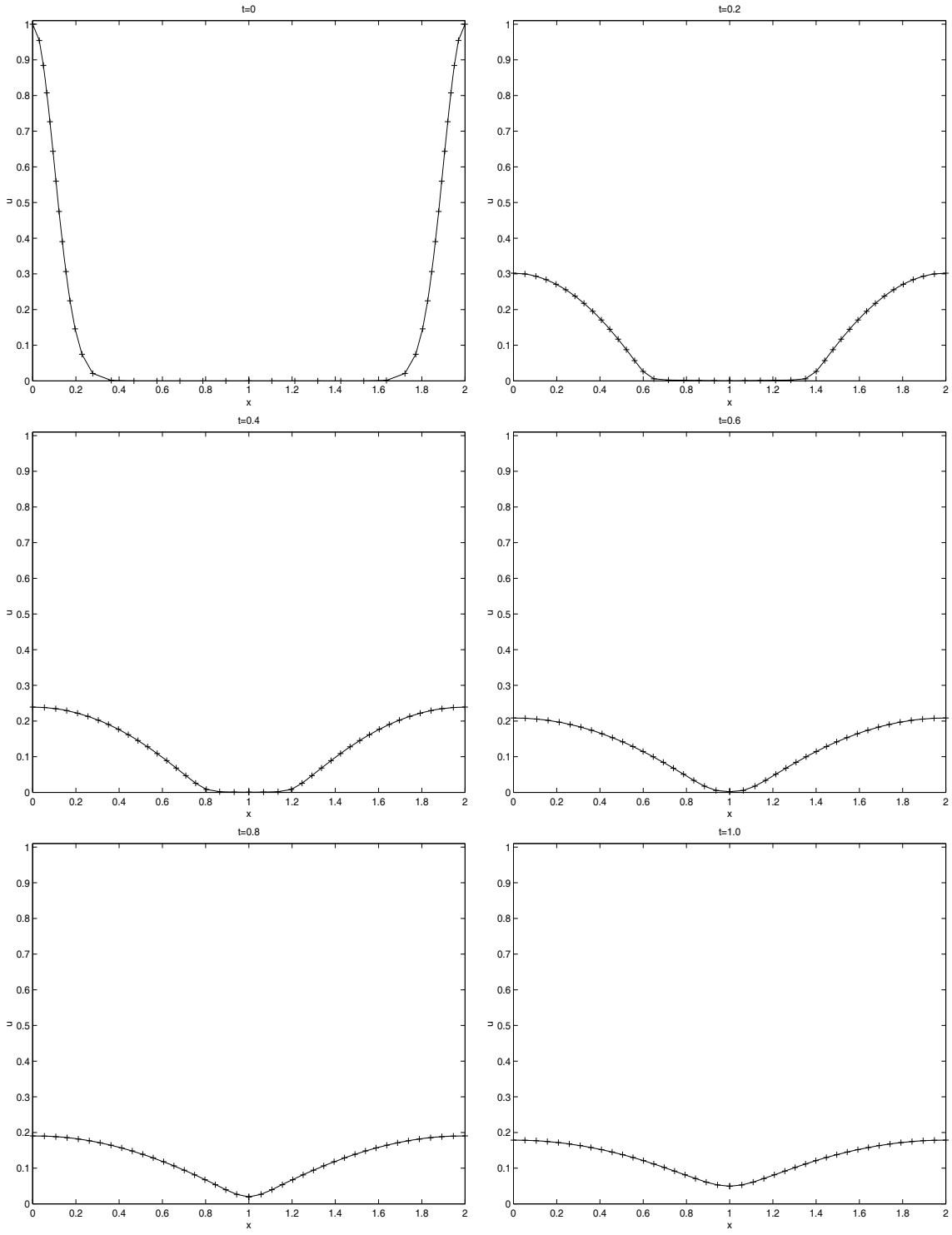

Figure 6.3: Approximate solution for the semiconductor problem with $\epsilon = 0.01$.

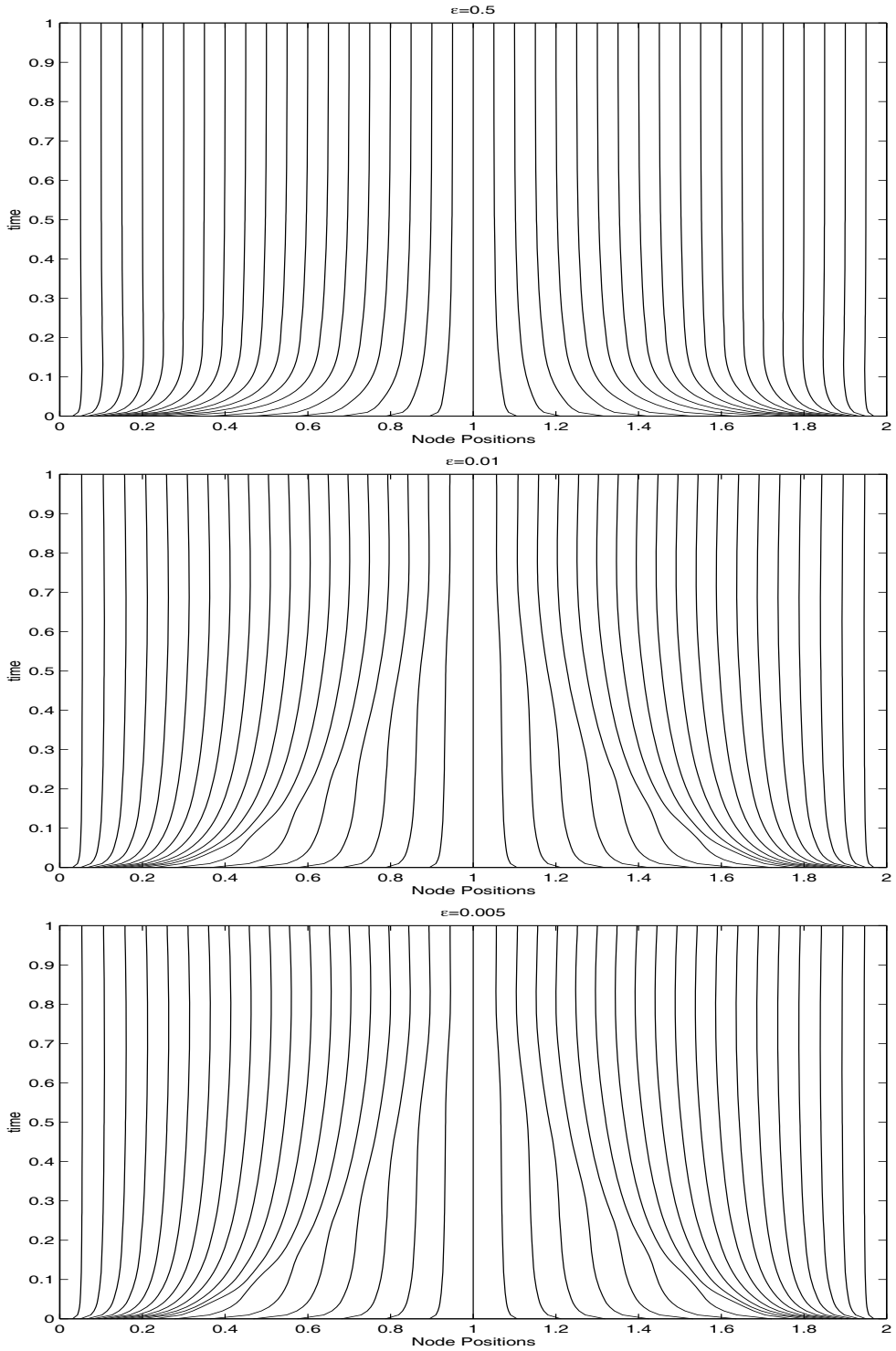

Figure 6.4: Node Trajectories for $\epsilon = 0.5, 0.01, 0.005$

6.2 Diffusion with Blow-Up

We now turn our attention to the solution of a PDE with blow-up. Such problems occur in physical models which develop singularities at some finite time T. Examples of such behaviour exist in the solution of equations describing combustion in chemicals or chemotaxis, with the blow-up possibly representing the ignition of a heated gas mixture. Budd, Huang and Russell [17] consider the following such problem of Fisher type,

$$u_t = u_{xx} + u^p \tag{6.10}$$

where $p > 1$.

A moving mesh method for this equation is suggested by Budd et al for two main reasons. Fixed mesh computations used to reproduce blow-up behaviour suffer from diminishing accuracy as the length scale of the singularity approaches the mesh spacing. For this reason such methods can often offer approximations which differ greatly from the underlying analytic solution, and it is reported in Budd that sometimes the stationary grid can miss the blow up in the solution entirely.

Equation (6.10), like the PME, is scaling invariant, in this case under the transformations

$$\begin{aligned} (T - \hat{t}) &\rightarrow \lambda(T - t) \\ \hat{u} &\rightarrow \lambda^{-\beta} u \\ \hat{x} &\rightarrow \lambda^{\frac{1}{2}} x \end{aligned} \tag{6.11}$$

for any positive constant λ with $\beta = \frac{1}{p-1}$. Since the singularity develops according to the same underlying structure, it is suggested that a moving mesh should be employed which also displays these properties. As with the PME, the moving mesh PDE can be made scaling invariant with a careful choice of monitor function. For equation (6.10) Budd et al, and more recently, Williams [76] suggest the monitor function

$$M(u) = u^{p-1}.$$

In this section will shall attempt to adapt the moving mesh method presented in Chapter 5 to generate solutions to equation (6.10). In particular we will be looking for the method to accurately reproduce the blow-up time T given in the literature to be approximately ≈ 0.082291, with the blow-up taking place at the origin in the form of an isolated spike of increasingly narrow width, developing from the initial and boundary conditions

$$u(x,0) = 20sin(\pi(\frac{1}{2} - x))$$

with

$$u(0,t)_x = u(1,t) = 0.$$

Considering the problem with $p = 2$, the monitor suggested by Budd et al corresponds to our earlier moving grid method using equidistributed mass, i.e. $M(u) = u$. However due to the inclusion of the source term the mass will be increasing with time and hence we must reconsider the time derivative of $\theta(t)$ term. By definition we have that

$$\begin{aligned} \theta_t &= \frac{1}{N-1}\int_{x_1}^{x_N} u_t dx \\ &= \frac{1}{N-1}\int_{x_1}^{x_N} (u_{xx} + u^2) dx, \\ &= \frac{1}{N-1}\left\{ [u_x]_{x_1}^{x_N} + \int_{x_1}^{x_N} u^2 dx \right\}. \end{aligned} \tag{6.12}$$

Moreover following the now familiar derivation of the moving mesh equations, beginning with

$$\frac{d}{dt}\int_{x_i(t)}^{x_{i+1}(t)} u dx = \theta_t \qquad 1, \ldots, N-1,$$

leads us to

$$(u\dot{x})|_{x_{i+1}} = \theta_t + (u\dot{x})|_{x_i} - \int_{x_i}^{x_{i+1}} u^2 dx - [u_x]_{x_i}^{x_{i+1}}.$$

We now have approximations to both the speeds of the nodes and the growth of the mass held in each cell. Our solution will be recovered using the fact that

$u(1,t) = 0$, using the trapezium relation from equation (5.7) in Chapter 5. However since our mass θ is now time dependent, we propose to add equation (6.12) to our ODE system and integrate forward in time. So we will then have the system variables

$$(\dot{x_1}, \dot{x_2}, \ldots, \dot{x_{N-1}}, \dot{x_N}, \theta_t)$$

using the discretisations

$$\begin{aligned} u_x &\approx \frac{u_i - u_{i-1}}{x_i - x_{i-1}} \\ \int_{x_i}^{x_{i+1}} u^2 dx &\approx \frac{1}{2}(u_{i+1}^2 + u_i^2)(x_{i+1} - x_i) \\ \theta_t &\approx \frac{1}{N-1}\left\{ \frac{u_N - u_{N-1}}{x_N - x_{N-1}} + \sum_{j=1}^{N-1} (u_{j+1}^2 + u_j^2)(x_{j+1} - x_j) \right\}. \end{aligned}$$

Figure 6.5 shows the resulting mesh when integrating forward implementing the suggested monitor $M(u) = u$. It is easy to see that the nodes do move towards the origin as the singularity develops. However, when plotting the solution for times near the end of the computations we see that the solution profile develops instabilities as the time of blow up approaches (Figure 6.6). The NAG routine was only able to integrate in time as far as $t \approx 0.067178$ as a result of the increasing instabilities in both the solution profile and the impending activity near the origin.

Although the monitor $M(u) = u$ preserves the inherent scaling properties in both the mesh and solution when we perform the resulting transformation (6.11) on the mesh, it appears that our moving mesh method has not, in this case, given us an invariant mesh in the transformed variables (Figure 6.5). Reasons for the developing instabilities are unclear, although it is obvious that their presence has an effect on the resulting mesh and the time at which the solution blows up.

In order to obtain a more satisfactory result, we propose a change to a different monitor function,

$$M(u) = \alpha - \omega u_x. \tag{6.13}$$

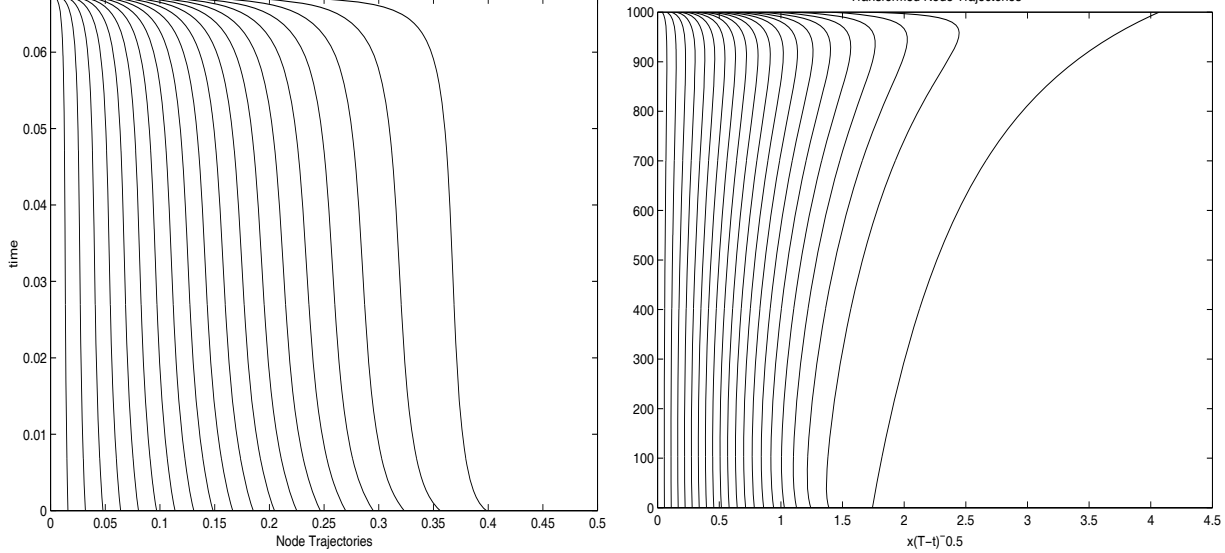

Figure 6.5: Trajectories for the blow-up problem with $M(u) = u$ (Left). Trajectories in transformed variables (Right).

It is anticipated that the presence of the first derivative will help to preserve monotonocity in the reconstructed solution, and moreover to provide fine mesh spacings as the solution approaches blow up and steep gradients develop. The additional constant term will allow us, as previously in Chapter 5, to preserve a reasonable mesh size in the solution near to the right-hand boundary in the time before blow up. This region seems quite important in determining the rate of growth in the mass (see equation (6.12)).

In deriving the appropriate equation, we still keep the same structure in our ODE system, replacing the θ_t term by an approximation to the time derivative of total mass $(TM)_t$ which will have a similar form to the replaced variable :

$$\begin{aligned}
(TM)_t &= \int_{x_0}^{x_N} u_t dx \\
&= \int_{x_0}^{x_N} (u_{xx} + u^2) dx \\
&= [u_x]_{x_0}^{x_N} + \int_{x_0}^{x_N} u^2 dx \\
&= u_x|_{\frac{1}{2}} + \int_{x_0}^{x_N} u^2 dx \\
&\approx \frac{(u_N - u_{N-1})}{(x_N - x_{N-1})} + \frac{1}{2} \sum_{j=1}^{N-1} (u_j^2 + u_{j+1}^2)(x_{j+1} - x_j) \qquad (6.14)
\end{aligned}$$

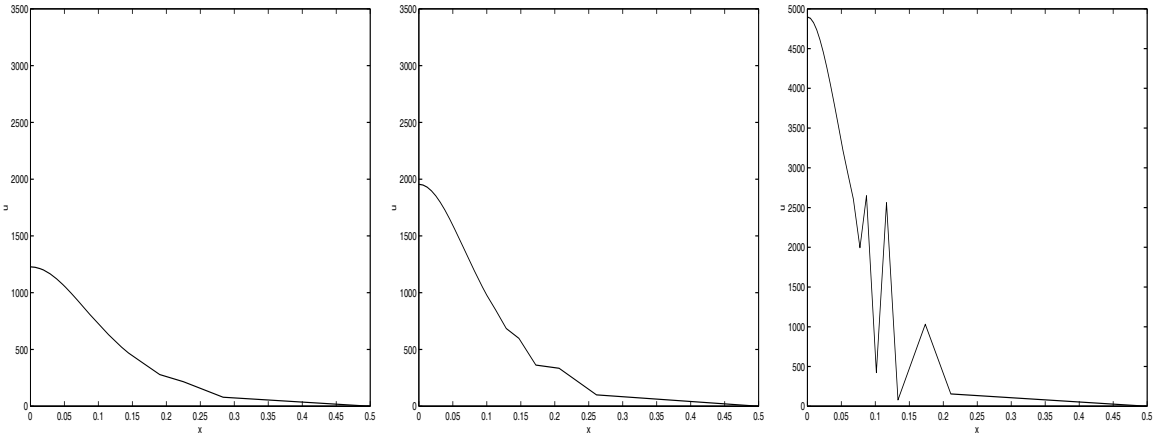

Figure 6.6: Approximate solution for times approaching blow-up.

The static equidistribution principle for the new monitor will provide approximations to the term $\theta(t)$ and hence the solution u from the current grid x by using a discrete approximation in the following manner,

$$\theta(t) = \int_{x_i}^{x_{i+1}} (\alpha - \omega u_x) dx \approx \alpha(x_{i+1} - x_i) - \omega \frac{(u_{i+1} - u_i)}{(x_{i+1} - x_i)}.$$

Rearrangement and further manipulation of this approximate relation over the domain gives us that

$$\begin{aligned} u_i &= \frac{\theta(t) - \alpha(x_{i+1} - x_i)}{\omega} + u_{i+1}, \\ &= 2\frac{\theta(t)}{\omega} - \frac{\alpha}{\omega}(x_{i+2} - x_i) + u_{i+2}, \\ &= \frac{\theta(t)}{\omega}(N - i) - \frac{\alpha}{\omega}(x_N - x_i). \end{aligned}$$

Substituting this expression into the trapezium expression (5.17) for the total mass TM gives

$$TM = \frac{1}{2} \sum_{j=1}^{N-1} (x_{i+1} - x_i) \left[\frac{\theta(t)}{\omega}(N - i) + \frac{\theta(t)}{\omega}(N - i - 1) - \frac{\alpha}{\omega}(x_N - x_i) - \frac{\alpha}{\omega}(x_N - x_{i+1}) \right]$$

which, when rearranged, gives an expression for $\theta(t)$,

$$\theta(t) = \frac{2TM - \Upsilon}{\Phi} \tag{6.15}$$

where

$$\Phi = \frac{\alpha}{\omega} \sum_{j=1}^{N-1} (x_{j+1} - x_j) \left[(x_N - x_i) - (x_N - x_{i+1}) \right]$$

and

$$\Upsilon = \frac{1}{\omega} \sum_{j=1}^{N-1} (x_{j+1} - x_j) \left[2(N-i) - 1 \right].$$

All that needs to be done now is to derive the moving mesh equations for the monitor (6.13). Differentiating the stationary equidistribution rule,

$$\frac{d}{dt} \int_{x_i}^{x_{i+1}} (\alpha - \omega u_x) dx = \theta_t. \tag{6.16}$$

and first considering the right-hand side term we have that

$$\begin{aligned} \theta_t &= \frac{1}{N-1} \int_0^{\frac{1}{2}} \alpha - \omega u_x dx, \\ &= \frac{-\omega}{N-1} \int_0^{\frac{1}{2}} u_{xt} dx, \\ &= \frac{-\omega}{N-1} \int_0^{\frac{1}{2}} u_{xxx} + (u^2)_x dx, \\ &= \frac{-\omega}{N-1} \left[u_{xx} + u^2 \right]_{x_1}^{x_N}. \end{aligned} \tag{6.17}$$

Returning to equation (6.16) and differentiating the left hand side gives us that

$$\begin{aligned} \int_{x_i}^{x_{i+1}} u_{xt} dx + \dot{x_{i+1}} (\alpha - \omega u_x)|_{x_{i+1}} - \dot{x_i} (\alpha - \omega u_x)|_{x_i} &= \theta_t, \\ -\omega \left[u_{xx} + u^2 \right]_{x_i}^{x_{i+1}} + \dot{x_{i+1}} (\alpha - \omega u_x)|_{x_{i+1}} - \dot{x_i} (\alpha - \omega u_x)|_{x_i} &= \theta_t. \end{aligned}$$

Rearranging and substituting into equation (6.17) we have the prescription for the mesh movement,

$$\dot{x_{i+1}} (\alpha - \omega u_x)|_{x_{i+1}} = \frac{-\omega}{N-1} \left[u_{xx} + u^2 \right]_{x_1}^{x_N} + \dot{x_i} (\alpha - \omega u_x)|_{x_i} + \omega \left[u_{xx} + u^2 \right]_{x_i}^{x_{i+1}}. \tag{6.18}$$

This is gives a set of equations to solve in conjunction with boundary conditions $\dot{x_1} = \dot{x_N} = 0$, along with the expression for the rate of growth of total discrete mass (6.14).

Applying the mesh algorithm with the new monitor function, we appear to generate a more satisfactory result. Our integration reaches a time of $T = 0.08155173$ which is much more reasonable compared to the time calculated approximately by Budd et al for blow-up. Moreover the value of u at $x = 0$ compares reasonably with a value of 10^{10}. The left hand side of Figure 6.7 shows that the mesh trajectories do not differ greatly from those using the mass monitor but the difference now is that our solution values stay monotonic. The change of monitor and the expression with which we recover the solution imposes that the solution always decreases away from the origin. Figure 6.8 shows the development of the solution as blow-up approaches, plotted with the solution divided by the maximum to provide a better understanding of what is happening. As can be seen, the nodes are moving towards the centre as the solution develops rapidly. To give an idea of the scale of the solution and the rapid growth, the right hand side of Figure 6.7 shows the maximum value of u with time.

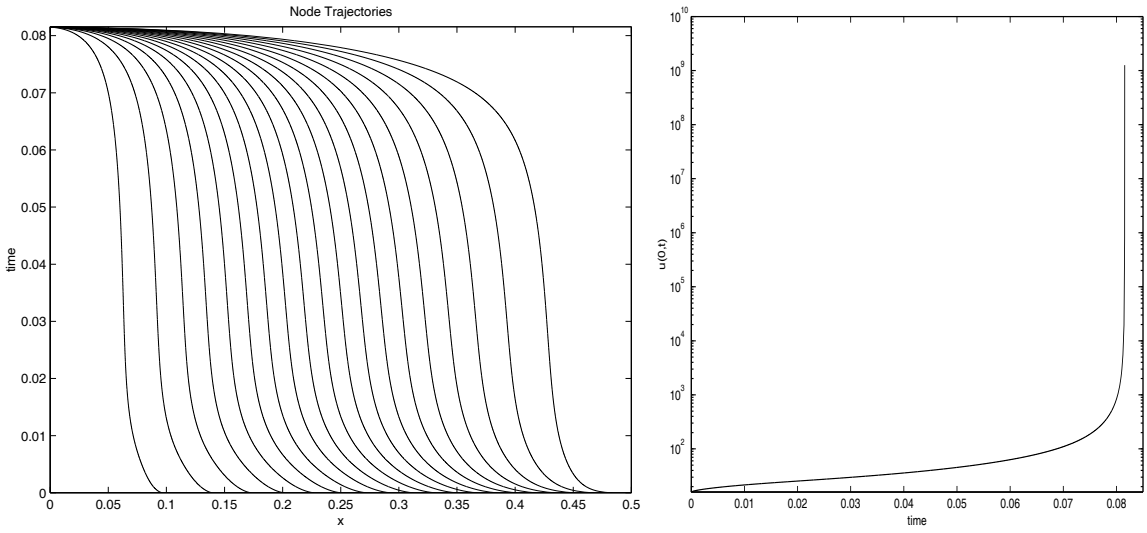

Figure 6.7: Mesh trajectories (Left) and evolution of $u(0,t)$ using a change of monitor function (6.13) with $\alpha = 1$, $\omega = 0.8$.

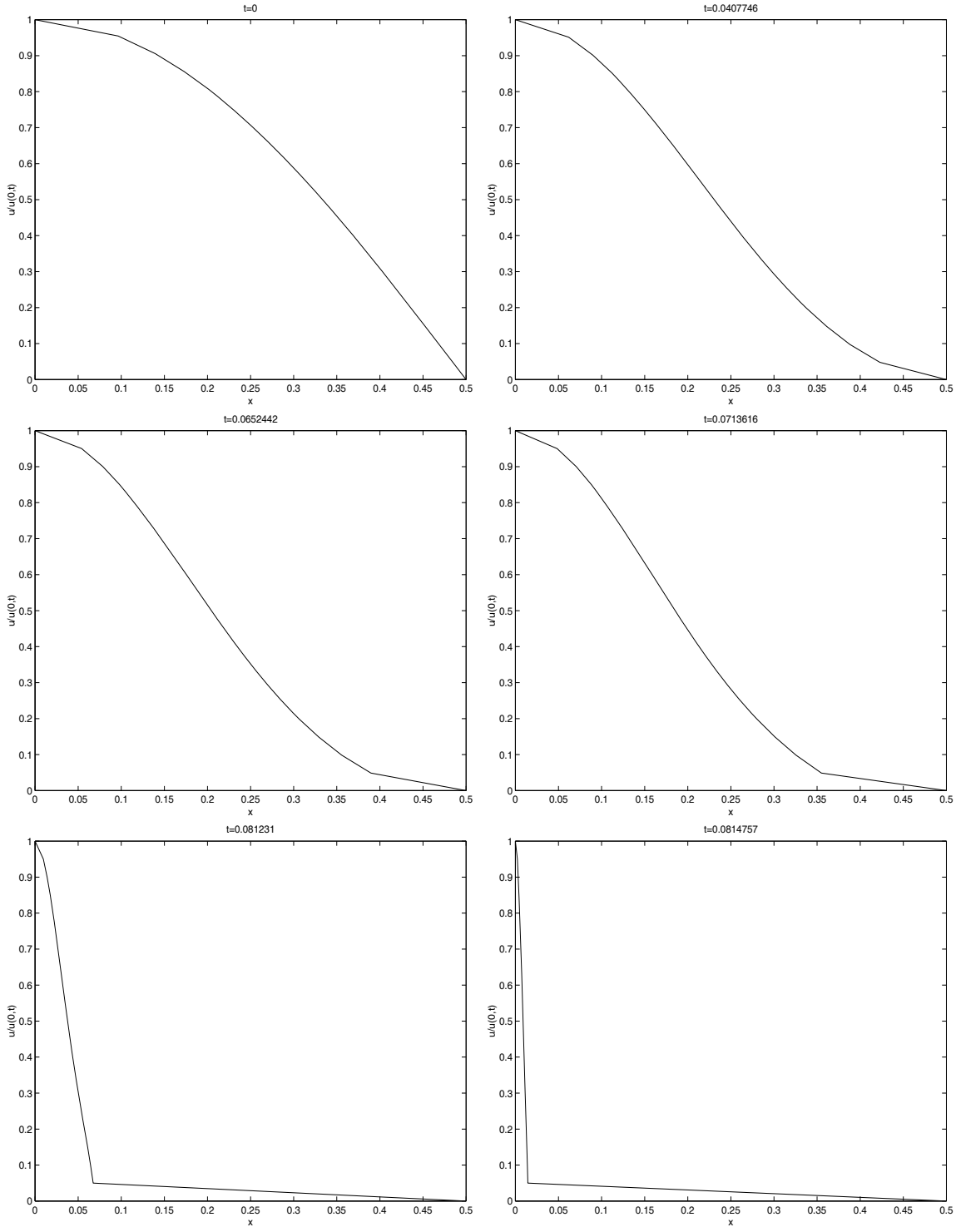

Figure 6.8: Approximate solution as blow up approaches, plotted as $\frac{u}{max_u}$

6.3 Summary

In this chapter we have attempted to illustrate that the moving mesh method derived in the previous chapter may be implemented on problems other than the PME. In the previous chapter, the derivation of the method relied heavily on taking advantage of the properties of the porous media solution and the aim was to produce similar moving meshes for problems without such inherent features, namely having a known solution value ($u = 0$) at a point and having a solution with constant mass. We have successfully adapted the method to include two applications without such useful characteristics.

The first of the problems was the now familiar semiconductor model problem. Here the difficulty was to be able to adapt to having a problem with an unknown solution over the entire mesh. This was overcome by including a local solution procedure at one boundary and deriving an approximation to the rate at which the solution develops at that point. The resulting term was then added to the ODE system and the global solution recovered using the current grid and the relevant quantity $\theta(t)$. Moreover, a combination type monitor was used to ensure a reasonable grid resolution around the point of the local solution update.

The remaining application was a diffusion problem with a singularity developing at the origin. After limited success with the invariant monitor suggested by Budd [17], we switched to a gradient type monitor. In this application mass is no longer conserved, so a term was derived to approximate the speed at which total mass was produced, added to the ODE system and integrated forward in time. Numerical results produced a time for blow-up in reasonable agreement with results produced by Budd.

Having extended the method to problems other than the PME whilst retaining the general theme of the method, i.e. solving for the mesh and then producing a reconstruction of the solution from the chosen monitor function, the next natural extension is to attempt a similar solution technique in higher dimensions.

Chapter 7

Two Dimensions

In the previous two chapters we have established an effective means of moving nodes for the solution of several PDE applications in one dimension. The underlying theme of the method is that correct evolution of the mesh is the priority with the solution being constructed, or even reconstructed, from the grid and a quantity $\theta(t)$ relating to the chosen monitor function specifying the grid movement.

Having achieved success in one dimension, the next natural step is to extend the method into higher dimensions. This chapter attempts to perform this transition to two-dimensional problems.

In one dimension we were able to adapt the method for several monitor functions and combination monitors. For simplicity we choose to use the simplest such monitor in this chapter, moving nodes via mass conservation. As with the work in one dimension, we develop this idea using the PME, and we shall make use of the Murray solution to the radially symmetric form of the PME. Although we are primarily interested in working with more general problems, this analytic solution allows us to evaluate our progress quantitively. As a precursor, we begin by extending the work in one dimension to solve the radially symmetric problem. We then move on, and attempt the generalized solution.

7.1 Solving the Porous Medium Equation Radially

An obvious approach to the radially symmetric porous medium solution is that of applying the one-dimensional techniques presented in Chapters 5 and 6 to the transformed problem in terms of the radial coordinate r. In doing so the transformed PME becomes

$$\frac{\partial u}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(ru^m\frac{\partial u}{\partial r}\right). \tag{7.1}$$

Once again a Neumann boundary condition is applied at the origin ($r = 0$). However, in the one-dimensional case, we took full advantage of the mass conservation properties of the equation. This conservation law, in radial coordinates is written as

$$\begin{aligned}
\frac{d}{dt}\int_0^{r_N(t)} rudr &= \int_0^{r_N(t)} r\frac{\partial u}{\partial t}dr, \\
&= \int_0^{r_N(t)} \frac{\partial}{\partial r}\left(ru^m\frac{\partial u}{\partial r}\right)dr, \\
&= \left[ru^m\frac{\partial u}{\partial r}\right]_0^{r_N(t)}, \\
&= 0.
\end{aligned}$$

It becomes apparent that we can use this conserved quantity as we did in the previous one-dimensional work. We now need to decide on a strategy for moving the mesh through the choice of monitor function. For simplicity we choose to move grid points in order to conserve discrete integrals of ru in each cell. From our experiences in one dimension it is obvious that for problems with higher powers of m in the diffusion coefficient that an initial equidistributed grid will not provide adequate grid resolution near the moving boundary. Hence we choose to generate an initial distribution using an ulterior monitor function and then to preserve the generated mass distribution of $\theta = ru$ with the evolution of the grid.

A simple effective form of monitor for the initial distribution in such a problem is likely to be that of

$$M(u) = 1 - \alpha u_r. \tag{7.2}$$

With a careful choice of α, an effective resolution at the steep front should be achieved whilst providing adequate resolution near to the origin.

Particular consideration needs to be paid to the discrete expressions representing the monitor functions involved in both the initial grid generation and the grid movement. Previously, all of the monitor functions have been represented by linear relationships between u and x, or in this case u and r. In this case, the mesh movement monitor ru is now a quadratic expression. Hence a more suitable numerical quadrature to take for the integral of ru over the cells is to use Simpson's Rule (which is exact for quadratics). In that case conserved integral $\theta_{i+\frac{1}{2}}$ over each cell $[r_i, r_{i+1}]$ will have the form

$$\int_{r_i}^{r_{i+1}} rudr = \theta_i \approx \frac{(r_{i+1} - r_i)}{6} \left[r_i u_i + 4 r_{i+\frac{1}{2}} u_{i+\frac{1}{2}} + r_{i+1} u_{i+1} \right]$$

where $r_{i+\frac{1}{2}} = \frac{1}{2}(r_i + r_{i+1})$.

Imposing a linear relationship between u and r allows us to express the central term in the above approximation in terms of the values at the endpoints on the cell, giving us that

$$\theta_i = \frac{(r_{i+1} - r_i)}{6} [r_i u_i + (r_{i+1} + r_i)(u_i + u_{i+1}) + r_{i+1} u_{i+1}] \tag{7.3}$$

which when rearranged leads to

$$\frac{6\theta_i}{2r_i + r_{i+1}} - \frac{2r_{i+1} + r_i}{2r_i + r_{i+1}} u_{i+1} = u_i. \tag{7.4}$$

We now have an algebraic relation between the current grid and the solution u. Equation (7.3) provides the quantities of $\theta_{i+\frac{1}{2}}$ to be conserved from the initial mesh, whilst its rearranged form gives the relationship between the current solution, u, $\theta_{i+\frac{1}{2}}$ and the grid r, given the boundary condition of the PME, $u_N = 0$.

We now turn our attention to the moving mesh equations. As a variation, we derive these expressions in a slightly different way from before, such that the speed of the moving boundary will be included within the general framework. Working on

a similar one-dimensional grid but in radial coordinates, we have the mesh consisting of N nodes, $r_0, \ldots, r_N$. Since the quantity ru is conserved over each cell, we have that

$$\frac{d}{dt}\int_0^{r_i(t)} ru\,dr = 0 \tag{7.5}$$

which, when expanded gives

$$\int_0^{r_i} ru_t\,dr + \dot{r}_i u_i r_i = 0. \tag{7.6}$$

Substituting the radial form of the PME (7.1) into the integral on the left-hand side gives

$$\begin{aligned}
\dot{r}_i u_i r_i &= -\int_0^{r_i} \frac{\partial}{\partial r}\left(ru^m \frac{\partial u}{\partial r}\right) dr, \\
&= -\left[ru^m u_r\right]_0^{r_i}, \\
&= -r_i u_i^m \left(u_r\right)_i .
\end{aligned}$$

We therefore obtain an expression for the speeds of all the nodes in the mesh. Upon further simplification, we have that

$$\dot{r}_i = -\frac{1}{m}\frac{\partial}{\partial r}\left(u_i^m\right). \tag{7.7}$$

As before, we use upwinding spatial discretisations for the derivative u_r, giving us the approximations

$$\begin{aligned}
\dot{r}_0 &= 0, \\
\dot{r}_i &= -\frac{1}{m}\left(\frac{u_i^m - u_{i-1}^m}{r_i - r_{i-1}}\right) \qquad i = 1 \ldots N,
\end{aligned}$$

to apply in the BDF NAG routine.

Figure 7.2 shows the results of this approach, taking $m = 3$ and $\alpha = 0.1$, showing good agreement with the analytical solution presented in Murray [64]. To show the effect of the parameter α in the initial distribution of ru over the mesh, Figure 7.1

shows two graphs. On the left hand side we see how the conserved discrete quantity of ru, as outlined in equation (7.3), changes with increasing values of α in the monitor function(7.2). The right-hand side of the figure illustrates how the changes in the initial refinement influence the error of the resulting time integration. The result as we would hope to see, is greater grid refinement near the front for larger values of α, providing us with a more accurate answer in terms of estimating the position of the moving boundary.

As with the results for the mass monitor in one dimension, we can verify our results by transforming the approximate mesh coordinates and solution values to the invariant solution variables using equation (2.44) in Chapter 2. Figure 7.3 clearly shows this invariant behaviour under the appropriate transformation in both the spatial and solution coordinates. This would suggest that our numerical method is indeed approximating the correct solution by using the mass conservation monitor in both the one-dimensional and radially symmetric problems.

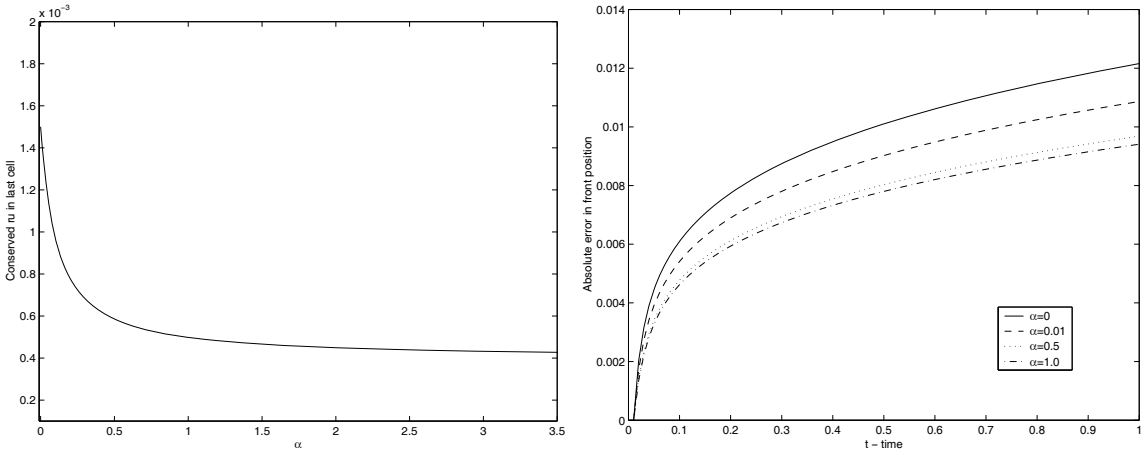

Figure 7.1: The quantity ru conserved in the last cell with the value of α in monitor (Left). Absolute error in front position for various values of α (Right).

This section has successfully computed the solution to this special case of the PME in twodimensions using a radial coordinate system. Although ultimately we wish to compute a solution within a more general higher-dimensional framework, by using the radial form of mass monitor i.e. $M(u) = ru$ we have further extended the range of monitor functions which can be implemented using this moving mesh idea. The next section will provide the basis of the higher-dimensional ideas that we seek.

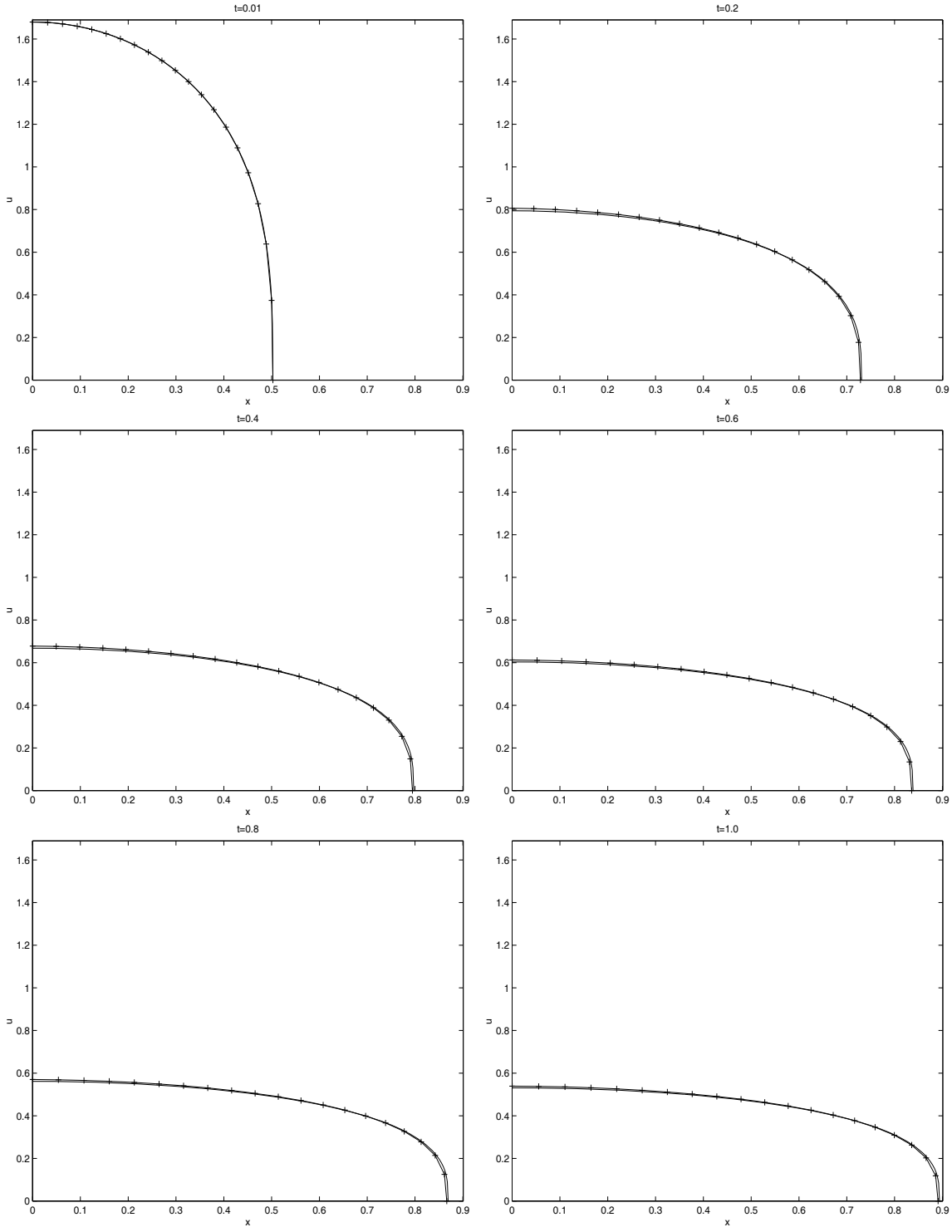

Figure 7.2: Radial Approx Solution to symmetric porous media problem in two-dimensions, $m = 3$, $\alpha = 0.1$.

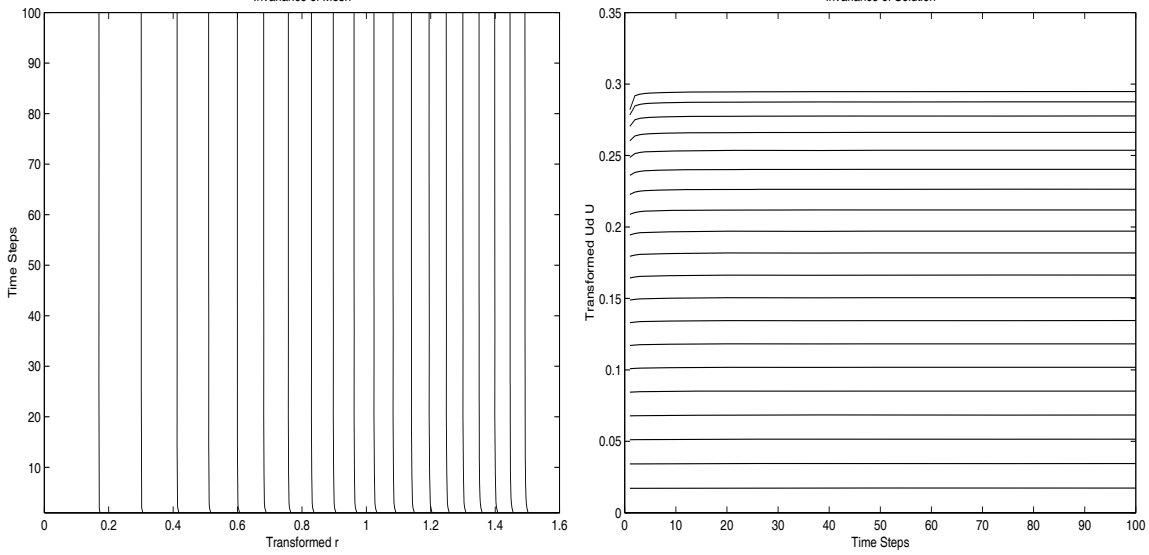

Figure 7.3: The invariant transformed computed radial coordinates (Left) and radially symmetric PME solution (Right).

7.2 A Two Dimensional Solution Approach

We have previously presented a solution for a radially symmetric problem above. However in most cases a problem in two or more spatial dimensions will not incorporate such a manageable geometry. We now present a solution strategy for more general problems.

We shall attempt to follow the methodology from the previous work undertaken in one dimension. It is sensible to consider the simplest approach to start with, in this case moving the nodes by conserving mass. In theory it seems possible to follow this approach directly into higher spatial dimensions. We shall attempt then to derive equations to continuously redistribute computational nodes by conserving discrete approximations to the volumes of diffusing material contained within each cell.

Hence we begin by deriving an equation to numerically account for the speeds of nodes contained in the resulting moving mesh. As before in Chapter 5, this is done with specific reference to the PME, here in its full two-dimensional form,

$$\frac{\partial u}{\partial t} = \nabla.(u^m \nabla u) \tag{7.8}$$

with boundary conditions $\nabla u = 0$ at the origin and $u = 0$ at the moving boundary.

To represent the geometry of the grid we propose to work on a triangular mesh. In particular we can make use of the grids used in Chapter 3, modifying them so that the moving boundary is followed exactly. For many problems, the technique of slicing the solution range equally should provide adequate mesh resolution near the steep moving boundary for greater values of m. To be precise, we propose to follow the methodology presented in Section 5.2. It makes sense to do this since to conserve equal measures of volume over the mesh would involve a highly accurate initial equidistribution process, requiring perhaps a functional minimising approach as in Baines [3], in view of the lack of a strict equidistribution principle in more than one dimension. Then, given an arbitrary domain Ω with boundary denoted $\partial\Omega$, by conservation of mass we have that

$$\frac{d}{dt}\int_\Omega u d\Omega = 0.$$

Expanding this integral gives a similar expression as in one dimension, with an integral for the rate of growth of mass inside the boundary in the fixed time frame and an integral compensating for the flux of mass out of the boundary.

$$\int_\Omega u_t d\Omega + \oint_{\partial\Omega} (u\underline{\dot{x}}).\underline{\hat{n}} dS = 0$$

where $\hat{n}$ represents the outward unit vector normal to the boundary $d\Omega$. Substituting (7.8) into the integral on the left-hand side, followed by the application of Gauss' divergence theorem, provides an equation for the prescribed velocities of grid points,

$$\oint_{\partial\Omega} (u\underline{\dot{x}}).\underline{\hat{n}} dS = -\oint_{\partial\Omega} (u^m \nabla u).\underline{\hat{n}} dS. \tag{7.9}$$

We shall be referring to equation (7.9) in the following sections where we shall derive a suitable discretisation and solution strategy for the moving mesh equation and also provide an expression for the speed of the moving boundary. In addition to having an equation prescribing mesh movement, we need a higher dimensional approach with which to extract a solution from the current mesh coordinates, through the conserved quantities of mass. The next section outlined three possible such approaches.

7.2.1 Recovering u from the Mesh

For continuity we would ideally like the update procedure to in some sense follow the work in one dimension. When considering the PME we have two boundary conditions, a Neumann condition at the origin and a Dirichlet property at the moving boundary, namely that $u = 0$. In one dimension our strategy was to enforce mass conservation in each cell via a numerical quadrature rule, imposing the Dirichlet condition to complete the resulting bi-diagonal system. In higher dimensions the completeness of the system is no longer achieved as easily. This section outlines three possible methods for constructing a solution over the mesh.

A Minimal Least Squares Solution

All approaches stem from imposing a discrete form of the mass conservation idea, namely that at all times a simple approximation to the quantity of mass held in each cell is unchanged as the mesh evolves. Hence we have that, over each triangle $\triangle_i$

$$\int_{\triangle_i} u d\triangle = \theta_i$$

where θ_i is the conserved mass. We approximate linearly giving

$$\theta_i = \frac{A_i}{3} \sum_{j=1}^{3} u_{i_j} \tag{7.10}$$

where u_{i_j} is the value of u at one of the three vertex's of triangle i.

In general the number of triangles in the mesh is greater than the number of nodes. It follows then that the resultant set of equations over the entire mesh will yield a non-square, overdetermined system, since we have more equations to satisfy than we have unknowns, i.e.

$$A\underline{u} = \underline{b} \tag{7.11}$$

where $A \in \Re^{p \times q}$ and $\underline{b} \in \Re^p$ and $p > q$.

The equation (7.11) cannot be expected to have an unique solution. We can however consider the use of a least-squares approach, which seeks a solution to

(7.11), u, such that

$$\|\underline{b} - A\underline{\bar{u}}\|_2 \tag{7.12}$$

is minimized. It is noted that for problems with full rank, the least squares solution is unique. However because of the connectivity's of the grids we work on and the impositions of $u = 0$ over the moving boundary, we have that

$$rank(A) = NC - 1$$

where NC is the number of cells in the grid (and hence number of equations in (7.11)), in which case there exist many possible $\underline{\bar{u}}$ which satisfy the least squares constraint.

As our system is over-determined, the singular value decomposition (SVD) method will find the solution $\underline{\bar{u}}$ which is in some sense minimal. To be precise, of all the solutions for which (7.12) holds, the method picks the particular $\underline{\bar{u}}$ for which $\|\bar{u}\|_2$ is as small as possible [73]. It is this solution that we choose to represent u over the mesh. We use the NAG routine $F04JGF$ [36] to compute our SVD solution, which computes the SVD solution irrespective of the rank of A. Moreover, when Dirichlet conditions are applied, the size of the system is reduced, although it only becomes less overdetermined.

A 'Patchwise' Approach

Our second attempt to construct an appropriate solution from the conserved masses and the current mesh collects together the masses in triangles around a single node, and requires them to remain constant. Doing so results in a square set of linear equations and avoids the need for a least-squares approach.

We now introduce a 'patch' of triangles associated with a single node i. The patch i is the collection of grid cells which contain node i as a vertex. Figure 7.4 illustrates the patch associated with the annotated node i, the outlined edges marking the boundary of the patch.

Obviously there is one patch for each node, so the idea is to form an equation relating the conserved mass and solution values over each patch. This is easily

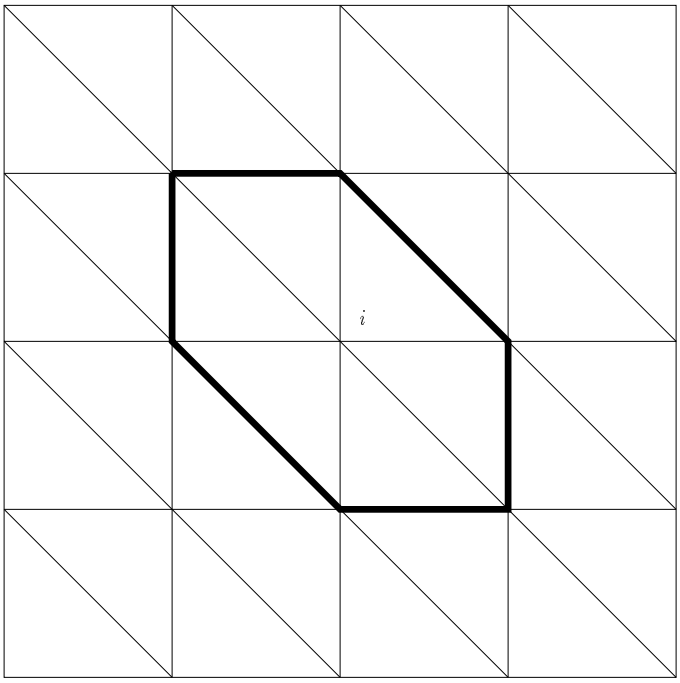

Figure 7.4: The patch of triangles associated with node i

achieved via summation, simply summing the equations (7.10) for each triangle contained within the patch giving us

$$\sum_{\triangle_j \in \mathrm{P}atch\mathrm{i}} \theta_j = \frac{1}{3} \sum_{\triangle_j \in \mathrm{P}atch\mathrm{i}} A_j \sum_{k=1}^{3} u_{j_k} . \tag{7.13}$$

This leaves us with a square system to solve for which any suitable matrix solver can be used. We choose to compute a solution via LU factorisation, which is undertaken via the NAG routine $F04ARF$ [36].

An Advancing Front Approach

Our second approach is much more in keeping with the philosophy of the one dimensional ideas, that being one of 'tracing' back the solution from the Dirichlet condition imposed on the boundary nodes. In one dimension, the numerical trapezium rule (or Simpson's rule used in the radial case) used to conserve mass over a cell is written in terms of two known current grid points, one known solution value and one unknown u_k, hence giving a unique solution for u over the grid. This allowed

us to work 'backwards' from the boundary and construct a solution over the mesh.

In two dimensions the extra degree of freedom in spatial coordinates means that our higher-dimensional mass conservation equation (7.10) cannot be imposed in such an ordered way. However, consider a triangle i and assume that the value of u is known at two of the cell vertices j and k, hence giving from (7.10) an equation for the solution at the remaining node l,

$$u_l = \frac{3\theta_i}{A_i} - u_j - u_k. \tag{7.14}$$

The imposing of the designated boundary condition on the moving boundary means that a sufficient number of cells which border the moving boundary will have the necessary conditions, upon which equation (7.14) may be called upon. Hence we can create a growing region of cells on which we can use this relation until all solution values are known. This 'advancing front' approach may then be implemented in the following way.

- Sweep through the mesh, until a cell is found with suitable criterion. i.e. two vertices with known solution values, one unknown.
- Using equation (7.14), calculate the value of u at the remaining vertex.
- Return to step 1 until u is known at all points in the mesh.

This approach may seem to be a more primitive approach when compared to the more robust methods stated previously. However, it does closely mirror the technique used in one dimension, and for this reason is deemed worth consideration.

Given a choice of these three strategies for satisfying the numerical quadrature relation over all cells, we can now return to the moving mesh equation (7.9) and derive a suitable discretisation and accompanying solution method.

7.2.2 A Moving Triangular Mesh

To begin, we reiterate the moving mesh equation derived (7.9) earlier. In order for cells to move such that mass is conserved within each cell, given an approximation to u over the current mesh, the velocities of the nodes $\underline{\dot{x}}$ should satisfy

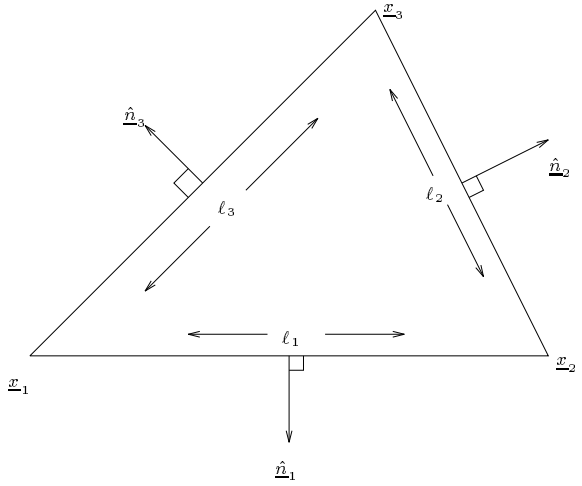

Figure 7.5: A general triangular mesh cell

$$\oint_{\partial\Omega} (u\dot{\underline{x}}).\hat{\underline{n}} dS = -\oint_{\partial\Omega} (u^m \nabla u).\hat{\underline{n}} dS.$$

In deriving an approximation to this relation, we consider a general triangular mesh cell with vertex coordinates $\underline{x}_1, \underline{x}_2, \underline{x}_3$ numbered anticlockwise. Figure 7.5 shows such a cell with labelled edge lengths ℓ_i and unit normal vectors $\hat{\underline{n}}_i$.

Taking our approximation u to be linear, ∇u is constant over each cell and u is linear along the edges of each triangle. Hence in equation (7.9) the left hand expression is quadratic and a Simpson's approach is used (as previously in section 7.1, whilst we retain a trapezium rule approximation for the left hand term. Hence we have

$$\sum_{j=1}^{3} \frac{\ell_j}{6} \left(u_j \dot{\underline{x}}_j + 4u_{j+\frac{1}{2}} \dot{\underline{x}}_{j+\frac{1}{2}} + u_{j+1} \dot{\underline{x}}_{j+1} \right) \hat{\underline{n}}_i = -\sum_{j=1}^{3} \frac{\ell_j}{2} (u_j^m + u_{j+1}^m) \nabla u . \hat{\underline{n}}_j, \qquad (7.15)$$

as an approximation to the moving mesh equation (7.9), where the summation is carried out in a cyclic manner anticlockwise around the triangle in consideration. The solution values u_i are the current approximation values given at the vertex's $\underline{x}_i$, whilst the constant gradient function ∇u is calculated over the i^{th} triangle via the formula

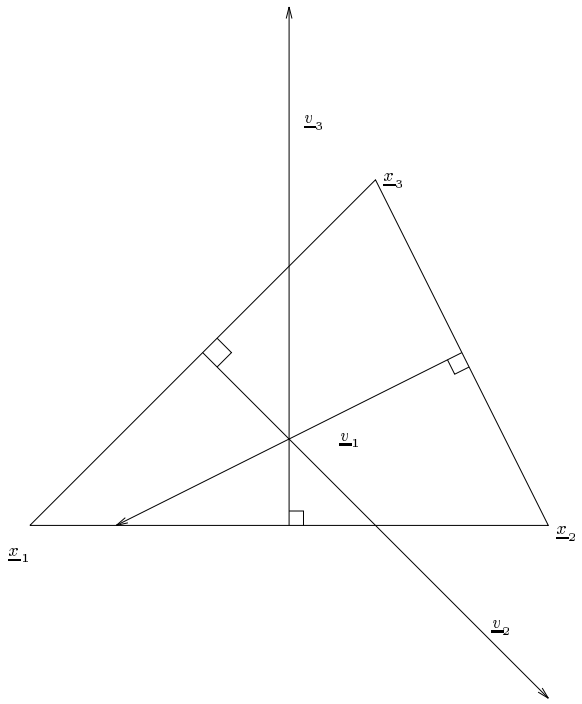

Figure 7.6: Calculating ∇u over the i^{th} triangle

$$\nabla u = \frac{1}{2A_i} \sum_{j=1}^{3} u_j \underline{v}_j \tag{7.16}$$

where $\underline{v}_j$ is the inward normal of the opposite edge to the vertex j as illustrated in Figure 7.6 and A_i is the area of the triangle i calculated by

$$A_i = \sqrt{s(s-\ell_1)(s-\ell_2)(s-\ell_3)}$$

where $s = \frac{1}{2}(\ell_1 + \ell_2 + \ell_3)$ ([57] & [51]).

Since we are trying to follow the previous work in one dimension, we also attempt to introduce some sense of upwinding in the approximation of ∇u. If we are considering integration over the edge j of the i^{th} triangle, we denote triangle k as 'outward' cell adjacent to edge j, (see Figure 7.7). Hence within equation (7.15) we choose ∇u according to

$$\nabla u = \begin{cases} \nabla u_i & \text{if } \nabla u_k . \underline{\hat{n}}_j \geq \nabla u_i . \underline{\hat{n}}_j . \\ \nabla u_k & \text{otherwise.} \end{cases}$$

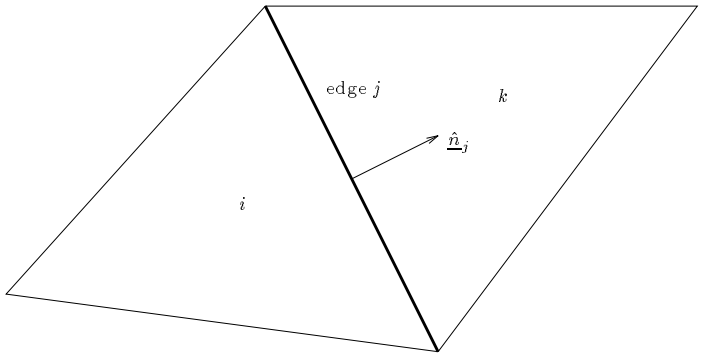

Figure 7.7: Choice of ∇u when integrating along edge j

We now have in place a moving mesh equation (7.15) with a suitable method of discretisation. However, by inspection of the resulting discretisation there is a similar discrepancy in the number of equations generated by each equation and in the total number of unknowns as for the u problem. The next section identifies a suitable transformation to improve the solution of the resulting non-square system.

7.2.3 Transforming to normal velocities

The set of equations resulting from equation (7.15) will however be underdetermined. This simply arises from the fact that node velocities now have two directional components. We are presented with NC equations (number of triangles) in $2NN$ unknowns (two velocity components for each node). Our grids are generated such that in general

$$NN < NC < 2NN. \tag{7.17}$$

Such a system has either no solution or an infinite number of solutions [35]. Since we need a unique solution to pass through to our ODE package for the grid, we require to treat the system in a way which can guarantee such a solution.

We shall define the normal velocity as the magnitude of the node speed in the direction normal to the contours of u. In the current system variables, if the normal velocity is $\dot{n}$, then the Cartesian form at node i, may be written as

$$\underline{\dot{x}}_i = \frac{\dot{n}_i}{|\nabla u|}\nabla u. \tag{7.18}$$

This suggests a suitable transformation, since by using $\underline{\dot{n}}$ instead of $\underline{\dot{x}}_i$, we then have only one unknown for each node i, hence from (7.17) we will have moved from an underdetermined to an overdetermined system, which can be guaranteed a solution via an SVD matrix computation.

Equation (7.18) requires information regarding the gradient of u at the grid points. However since our representation of the solution is piecewise linear, this cannot be done exactly. We therefore propose the approximation to ∇u at node i, denoted $\overline{\nabla} u$, using an averaging of ∇u calculated over the triangles belonging to the patch associated with node i, i.e.

$$\nabla u_i \approx \overline{\nabla} u_i = \frac{1}{N} \sum_{\triangle_j \in \, patch\, i}^{N} \nabla u_j \tag{7.19}$$

where N is the number of triangles in patch i.

Hence combining equations (7.19) and (7.18) we have at each node transformations to the normal velocity $\dot{n}_i$ from the x and y components of $\underline{\dot{x}}_i$,

$$\dot{x}_{i_x} = \frac{\overline{\nabla} u_x}{\left|\overline{\nabla} u\right|}\dot{n}_i \qquad \text{and} \qquad \dot{x}_{i_y} = \frac{\overline{\nabla} u_y}{\left|\overline{\nabla} u\right|}\dot{n}_i \tag{7.20}$$

where $\overline{\nabla} u_x$ and $\overline{\nabla} u_y$ are the x and y components of $\overline{\nabla u}$ respectively. As explained above, this transformation applied to the moving mesh equations (7.15) changes the shape of the matrix system to be solved for the node speeds, giving

$$\hat{A}\underline{\dot{n}} = \underline{\hat{b}}. \tag{7.21}$$

Figure 7.8 illustrates the effect of the transformation on the dimensions of the matrix A and vector $\underline{b}$. In this form the system is guaranteed a least squares minimal solution via the SVD method, irrespective of the rank of A.

Moreover due to the relation between the number of triangles and nodes given in our grid (7.17), the matrix system will actual be smaller in terms of the total number of elements, hopefully making computations more efficient.

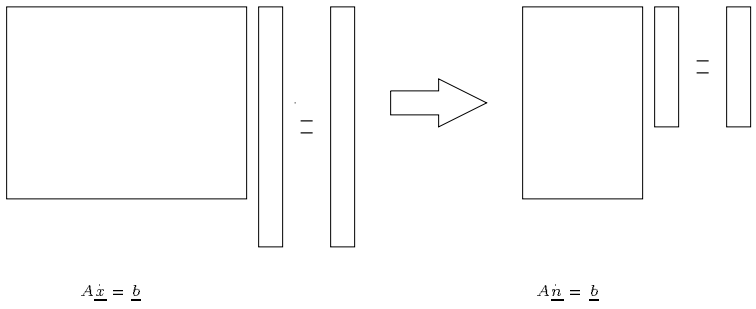

Figure 7.8: Effect of velocity transformation on the shape of matrix system

Following the solution of the resulting matrix equation, our speeds are transformed back to the original Cartesian form, using (7.18), and fed into the NAG backwards differentiation formula routine, as used in one dimension and integrated forward in time.

Finally, the resulting matrix system for the normal velocities requires a boundary condition specifying the speed of the moving boundary. When we considered the problem in the radial coordinate in Section 7.1, the speed of the moving front was included in this radial framework. However, since we wish to provide a more general solution, we need to provide an approximation to this speed, as previously in one dimension. It is obvious from (7.15) that since $u = 0$ on this section of the grid, the coefficients of these velocities in the matrix system (7.21) will be zero and may never appear in the resulting solution. The following section derives an approximation to the moving boundary in the transformed normal velocity coordinates.

7.3 The moving boundary

To complete the ODE system, as in one dimension, we provide a Dirichlet condition in the form of an approximation to the speed of the moving boundary. This section briefly covers the derivation of this approximation in two dimensions.

Figure 7.9 illustrates the domain Ω for the PME. We consider the mass conservation law over this quarter section of the entire circular domain

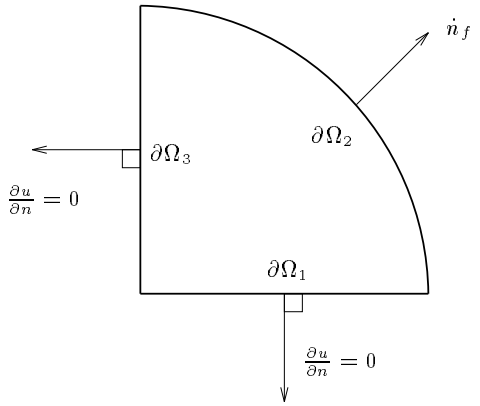

Figure 7.9: Diagnostic of the domain and boundary of the radially symmetric PME

$$\frac{d}{dt}\int_{\Omega} u d\Omega = 0.$$

Expanding the integral and using Gauss' theorem leaves us with a line integral around the domain boundary $\partial\Omega$. As illustrated in Figure 7.9, we divide the boundary into three separate component, $\partial\Omega_1$, $\partial\Omega_2$ and $\partial\Omega_3$. Thus

$$\oint_{\partial\Omega_1} \left((u^m \nabla u) + u\dot{x}\right).\underline{\hat{n}} dS + \oint_{\partial\Omega_2} \left((u^m \nabla u) + u\dot{x}\right).\underline{\hat{n}} dS + \oint_{\partial\Omega_3} \left((u^m \nabla u) + u\dot{x}\right).\underline{\hat{n}} dS = 0.$$

The line integrals over the first and last components of the boundary disappear, due to the symmetry boundary conditions on u and resulting node velocities over these section of the boundary. For the remaining integral we use the radial symmetry of the problem, that the solution and hence the magnitude of the velocity of the moving boundary in the normal direction, $\dot{n}_f$, will also be constant over this integral. Hence we have that

$$u\dot{n}_f + u^m \nabla u = 0. \tag{7.22}$$

By direct analogy with this procedure in one dimension in Section (5.1.3), u tends to zero and ∇u tends to infinity as the radial coordinate of $\underline{x}$ tends to the position of the moving front. However in a similar style as before we avoid this technicality by writing (7.22) as

$$\dot{n}_f = -\frac{1}{m}\nabla(u^m).\underline{\hat{n}}. \tag{7.23}$$

We compute this speed in an arbitrarily chosen cell which has an edge aligned with the boundary $\partial\Omega_2$, with u measured at the centre of the chosen cell.

Our method is now complete; all sections are put together, again utilizing the NAG BDF routine. We now present findings from the numerical computations.

7.4 Numerical Results VI

Initially we shall only be concerned with solving the PME in two dimensions for the simple case of $m = 1$. For now we shall be concerned with the evolution of the mesh and the resulting construction of the current solution. To provide a thorough treatment of the methodology presented above, we test the algorithm, in turn incorporating all three strategies of recovering u from the updated mesh. The grid is moved by the solution of the overdetermined system (7.21), which is solved for the normal velocities, which are in turn transformed back to mesh speeds in regular Cartesian form. The resulting ODE system is then integrated forward in time using the backwards differentiation formula NAG routine $D02EJF$ as used with all the work in one dimension.

To test the method further we run the three different routines on three meshes of increasing resolution in both the radial direction and along the contours on the meshes. The first grid is a very basic grid containing only 21 nodes and 28 triangles, the second has 82 nodes making up 136 elements, whilst the last grid incorporating 324 nodes and 592 cells. Following the work in Chapter (5), we take initial conditions arising from the Murray solution (2.40) at an arbitrarily small time $tstart = 0.01$. Figure 7.12 shows all three meshes and the associated initial conditions.

The BDF routine was set to stop the calculations at time $t = 1.0$. Unfortunately, few of the calculations reached their intended destinations. The BDF routine requires a user-defined error tolerance included in a choice of an adaptive time step. Upon using three different choices of this tolerance, it seems that the algorithm only integrates successfully to time $t = 1.0$ for simple, low resolution meshes in

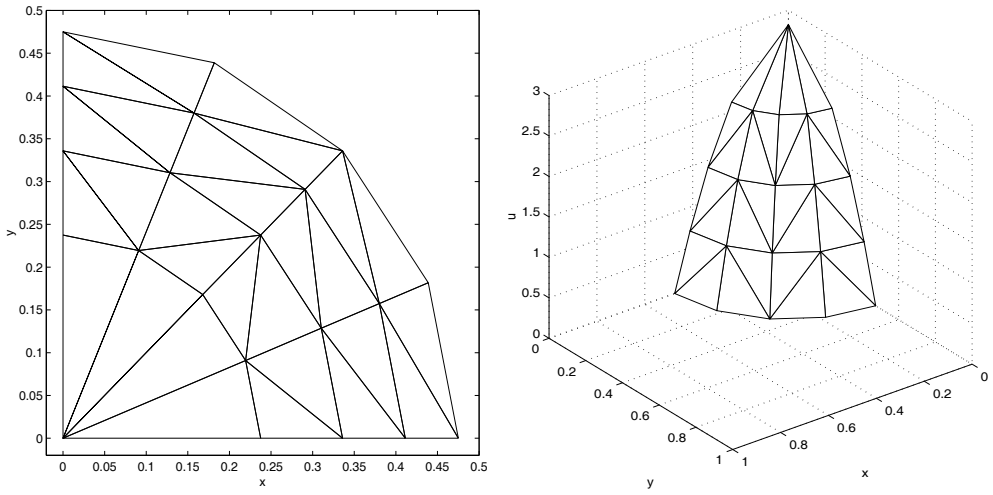

Figure 7.10: Mesh 1 : 21 nodes, 28 cells.

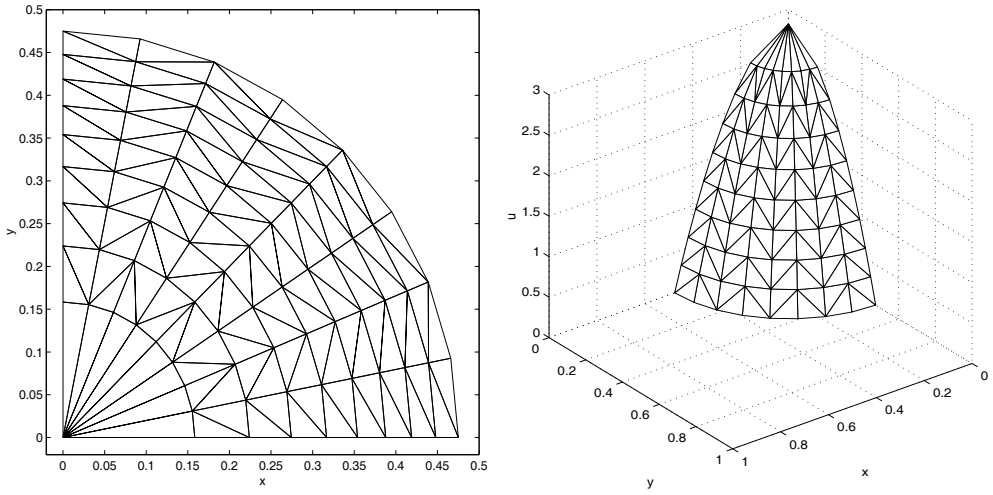

Figure 7.11: Mesh 2 : 82 nodes, 136 cells.

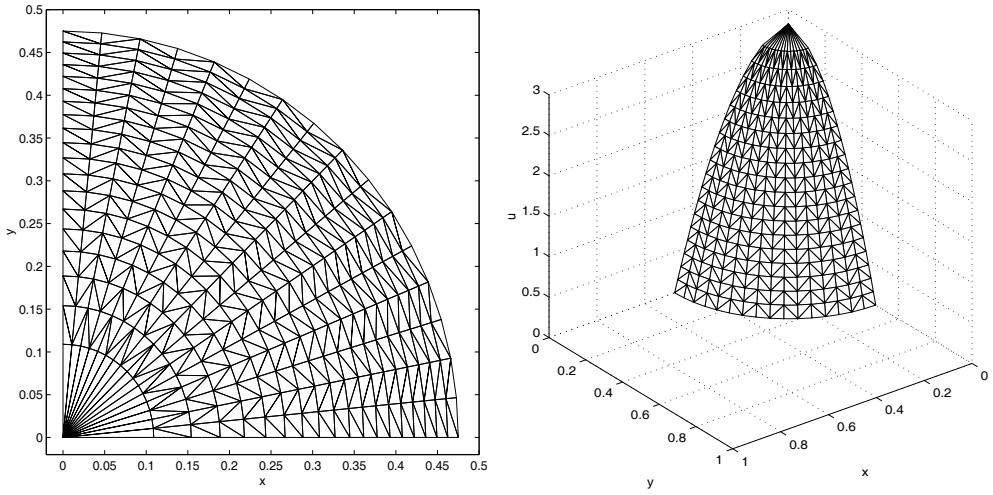

Figure 7.12: Mesh 3 : 324 nodes, 592 cells.

conjunction with a low tolerance level. For example, all three styles of u solution only manage to complete the integration with a tolerance of 10^{-3} on mesh 1. Tables 7.1 to 7.3 show the times at which the routines stopped. For the higher chosen tolerances the BDF struggles to integrate far past the initial time, the number of iterations tabulated illustrating the effort put into the computations.

Luckily we can study the two results successfully produced on mesh 2 to try and understand why the time integration is proving troublesome. Both the SVD and patchwise approaches for recovering u managed to integrate fully to time $t = 1.0$ and are able to plot the resulting grids and solutions at intermediate times. The two methods of constructing a solution from the mesh 2 do not give the same solution. They do however both exhibit an oscillatory nature in u. Figures 7.14 and 7.15 show the development of the approximate solution in these cases for the SVD and patchwise solutions respectively. It is easy to see at early times signs of instability or non-monotonic behaviour of the solution, most notably in the radial direction. In both cases the low tolerance seems to play a part in allowing the mesh to evolve forward in time, and it is worth noting in Tables (7.1-7.3) that when the routines stopped prematurely with the higher tolerances they generally failed to integrate past the initial times, when the outward motion of the mesh is at its fastest.

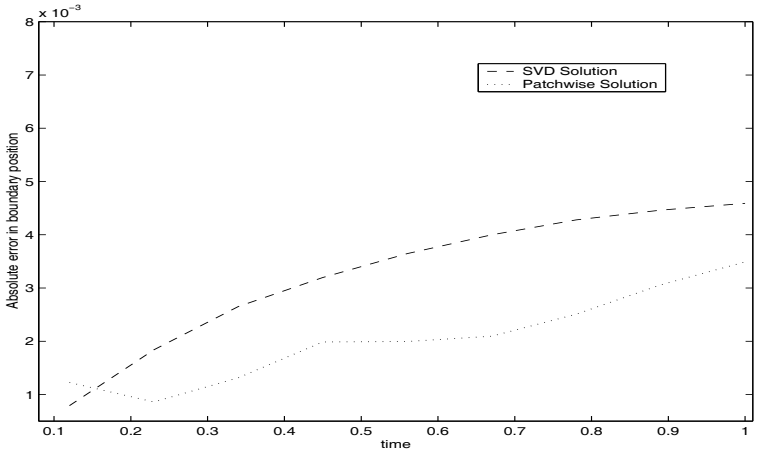

Figure 7.13: Absolute error in radial coordinate of moving boundary associated with Figures (7.14) and (7.15).

Despite the evidence of oscillatory behaviour appearing in the solution, the resulting approximations appear to have a level of accuracy in accordance with the

chosen tolerance. Figure 7.13 shows the error in radius of the moving boundary with time for both the SVD and patchwise solutions and despite the oscillations both remain below the specified tolerance.

Of the three solution approaches, the advancing front solution proved to be least successful. Although the equations involved in all computations are the same, this approach failed in all but one attempt to do the full time integration required. The source of apparent inaccuracy lies in the less rigorous manner in which the grid is swept for suitable cells and hence information is sometimes not used, information which the SVD and patchwise approaches do not ignore. Put simply, when the advancing front solution is complete, the number of triangles visited will be equal to only the subset of points with unknown solution values. From the relation stated earlier (7.17), it is obvious that the advancing front approach only considers a subset of the total triangular elements in the mesh. Since there is an equation relating conserved mass, cell area and u for each cell, the advancing front solution cannot be guaranteed to satisfy all of these equations. From this point on then, we shall disregard the advancing front approach.

The results for the chosen low tolerance are reasonably good and in some senses the method as it stands could be seen to be successful, in the sense of the direct translation of the approach from one to two dimensions. However, the failure of the BDF routine to be able to integrate forward in time for more reasonable tolerances and for higher resolution grids gives some cause for concern and needs further investigation (it should be pointed out that in one dimension significantly higher tolerances were used in the BDF NAG routine). Small time approximate solutions exhibit signs of instability somewhere in the system solved for $\underline{\dot{x}}$. With this in mind, the next section attempts to identify which part, or parts, of the algorithm admit these effects.

7.4.1 The Porous Medium Equation as a Special Case

In the previous section, we found limited success in applying the moving mesh idea to the PME in two dimensions. It was concluded that instabilities or oscillations were being produced in certain aspects of the method, these effects thought to have

	tol=10^{-3}		tol=10^{-7}		tol=10^{-11}	
mesh1	19030	1.0	11514	0.010687	9440	0.010764
mesh2	6536	1.0	56005	0.017427	40231	0.011818
mesh3	18293	0.234598	203341	0.0213211	126914	0.027653

Table 7.1: Success of BDF integration using the SVD solution for u. Shows time at which integration stopped and number of iterations taken for different tolerance and meshes.

	tol=10^{-3}		tol=10^{-7}		tol=10^{-11}	
mesh1	21052	1.0	6860	0.010647	12637	0.010568
mesh2	33172	1.0	27398	0.012639	46079	0.013468
mesh3	47293	0.0182345	96328	0.015642	148172	0.029423

Table 7.2: Success of BDF integration using the patchwise solution for u. Shows time at which integration stopped and number of iterations taken for different tolerance and meshes.

	tol=10^{-3}		tol=10^{-7}		tol=10^{-11}	
mesh1	2845	1.0	3364	0.015123	7162	0.011576
mesh2	113765	0.016492	52247	0.033169	29942	0.033187
mesh3	305896	0.019234	109824	0.042343	150192	0.052169

Table 7.3: Success of BDF integration using the advancing front approach for u. Shows time at which integration stopped and number of iterations taken for different tolerance and meshes.

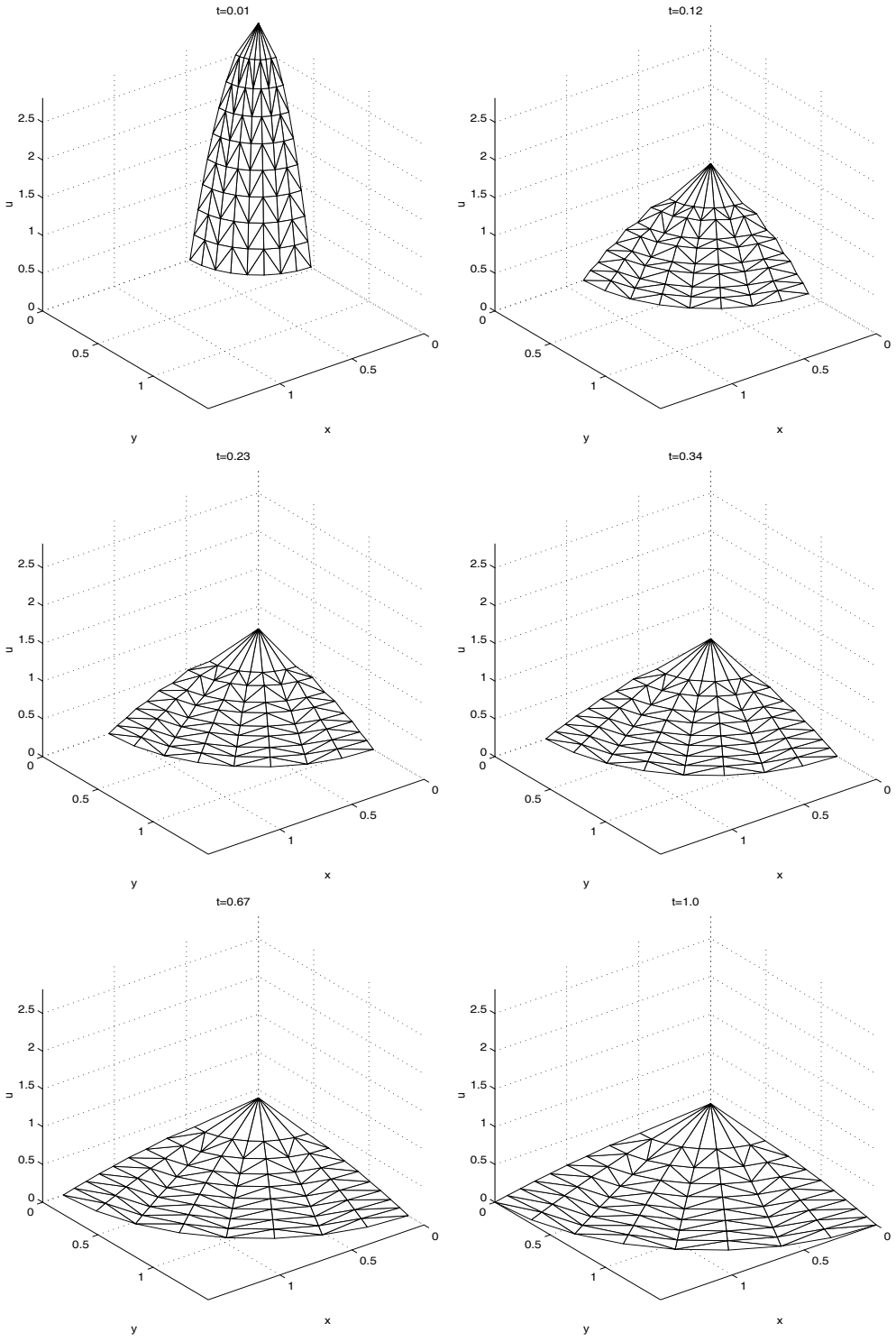

Figure 7.14: Approximate solutions ($m = 1$) using the SVD solution. Calculations using a tolerance = 10^{-3} on mesh 2.

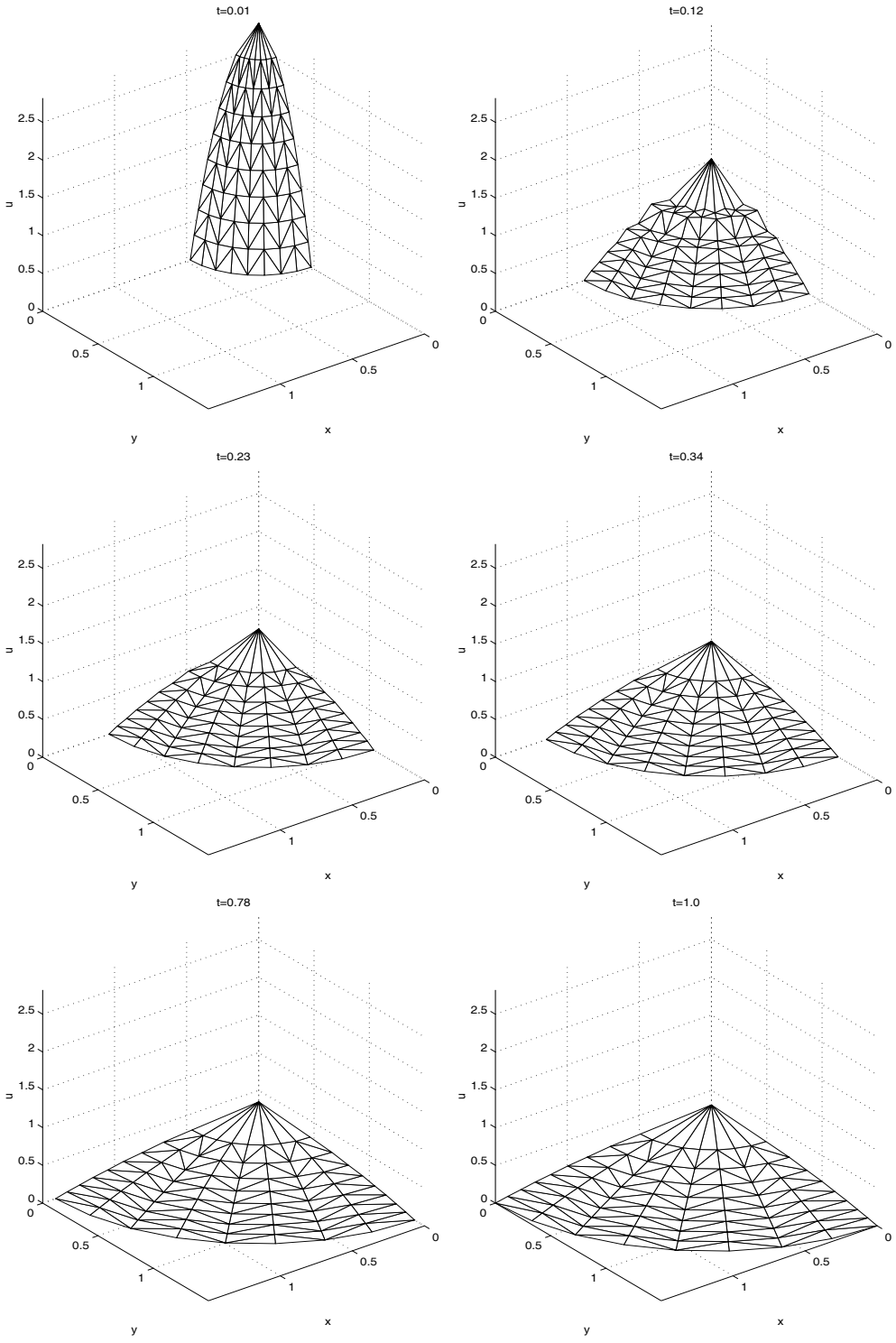

Figure 7.15: Approximate solutions ($m = 1$) using a 'patchwise' approach. Calculations using a tolerance $= 10^{-3}$ on mesh 2.

been produced early in the integration, when the velocities of the grid were at their peak. Only the use of relatively low error tolerances and low resolution allowed the approximate solution to overcome these difficulties to produce a reasonably accurate solution.

The aim of this section is to investigate the cause of these problems. We are mainly concerned with the two overdetermined systems used to deduce the correct node velocities from the current grid and solution. It was noted previously that, due to the connectivity style incorporated in the grid generation, these two systems cannot be guaranteed to have a unique solution, and the SVD routine chooses the least squares minimal solution. Hence we cannot be certain, as we were in one dimension, that all the equations regarding the conservation of mass are being exactly satisfied.

Fortunately the geometry of the radial PME and our current grids allow us to eliminate one of these systems. Since the grids in use are based upon contours of the solution u, we would expect, and have found in the previous section, that the grid would retain this characteristic with the nodes moving radially outwards. Indeed it could be a measure of success of the method as to how well the nodes along a contour propagate uniformly together. Section (7.3) identifies an approximation to the normal velocity of the moving boundary, and in the same way we can think of each 'contour' in the same way, moving by conserving the mass contained between the contour itself and the vertical and horizontal boundaries. Following this thinking, the approximation to the normal velocity of the moving boundary (7.23) is valid for all nodes inside the expanding domain too. However we do not wish to impose too many radial restrictions on the movement of the nodes inside of the boundaries, since we would be in effect solving the problem in radial coordinates. Hence we work in Cartesian components of velocity, with

$$\underline{\dot{x}} = -\frac{1}{m}\nabla(u^m) \tag{7.24}$$

regarded as an exact solution of the moving mesh equation,

$$\oint_{\partial\Omega}(u\underline{\dot{x}}).\underline{\hat{n}}dS = -\oint_{\partial\Omega}(u^m\nabla u).\underline{\hat{n}}dS.$$

Taking advantage of this special case allows us to derive the mesh speeds directly from the current solution u, temporarily bypassing the least squares solution of the system (7.21). Hopefully this process of elimination will shed some light on the issue of the source of instabilities currently present in the method. Equation (7.24) is approximated easily, by simply constructing the constant gradient of u^m over each cell and then taking a patchwise approximation to the speeds at each node using methods outlined earlier (see equations (7.16) and (7.19)).

Having disregarded the advancing front approach for the u solution, we apply the remaining two strategies in conjunction with equation (7.24) in approximating the correct mesh speeds. The correct form of the method allows us to enforce a much tighter error tolerance in the BDF routine. Once again both method successfully integrate forward in time till $t = 1.0$ on mesh 1. More interestingly, when calculations are undertaken on mesh 2 with the lower tolerance ($= 10^{-12}$), the methods are both successful in completing the integration forward in time. Moreover the signs of instability at early times found previously, and featured in Figures 7.14 and 7.15, have been eliminated. Figure 7.17 illustrates the now smooth evolution of the solution generated using the patchwise solution approach. The SVD solution produced a similar result, interestingly enough requiring more iterations (4676) in the NAG routine than solving patchwise (2889).

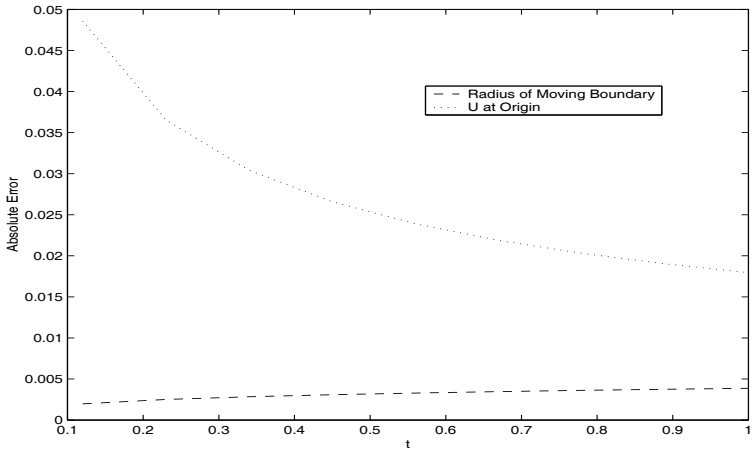

Figure 7.16: Error with time in patchwise solution with exact form of mesh speed. Calculations using a tolerance $= 10^{-12}$ on mesh 2.

Figure 7.16 shows two error measures associated with the plots in Figure 7.15.

The error in radius of the moving boundary shows a similar behaviour and reasonable accuracy as before, although there is clear room for improvement in the error in u at the boundary. Table 7.4 shows the approximate solution of both methods at various times as a comparison. Both the radial coordinate of the moving boundary and the solution at the maximum are tabulated. In both cases the solutions are generally within 4 decimal places of one another. These results suggest that the exact form of the node velocities considerably affects the method in the way that the BDF routine is now allowed to complete the time integration with a lower error tolerance. We can probably say then that this form has improved the solutions which were previously generated when solving the overdetermined system for $\underline{\dot{n}}$. We now present two cases that illustrate that the instabilities found previously still exist.

	U at Max.		Radius of Bdry.	
Time	SVD	Patchwise	SVD	Patchwise
0.12	0.86294386	0.86291238	0.88218906	0.88220434
0.34	0.51422315	0.51420394	1.14426904	1.44291873
0.56	0.40086826	0.40085327	1.29622135	1.29624663
0.78	0.33972117	0.33970803	1.40813309	1.40816094
1.00	0.30005901	0.30004741	1.49834917	1.49837887

Table 7.4: Comparison of SVD and Patchwise solutions at the maximum and in radial coordinate of moving boundary

Previously, in Section 7.4, Tables (7.1-7.3) showed that when the least squares solution for the normal velocities was used in the BDF routine the integration failed for mesh 3 irrespective of which solution method for u was used. We can assume that this was caused by instabilities in one or both of the solution stages. Figure 7.18 shows the evolution of solution when working on a grid with a slightly higher resolution than mesh 2, containing 109 nodes, using the same parameters with the SVD routine and the exact form of the mesh speeds. In this case the BDF routine fails to integrate past $t \approx 0.1210286$, but we can plot the solution up until this point. Initially the solution evolves smoothly as expected, but at the point of the failure

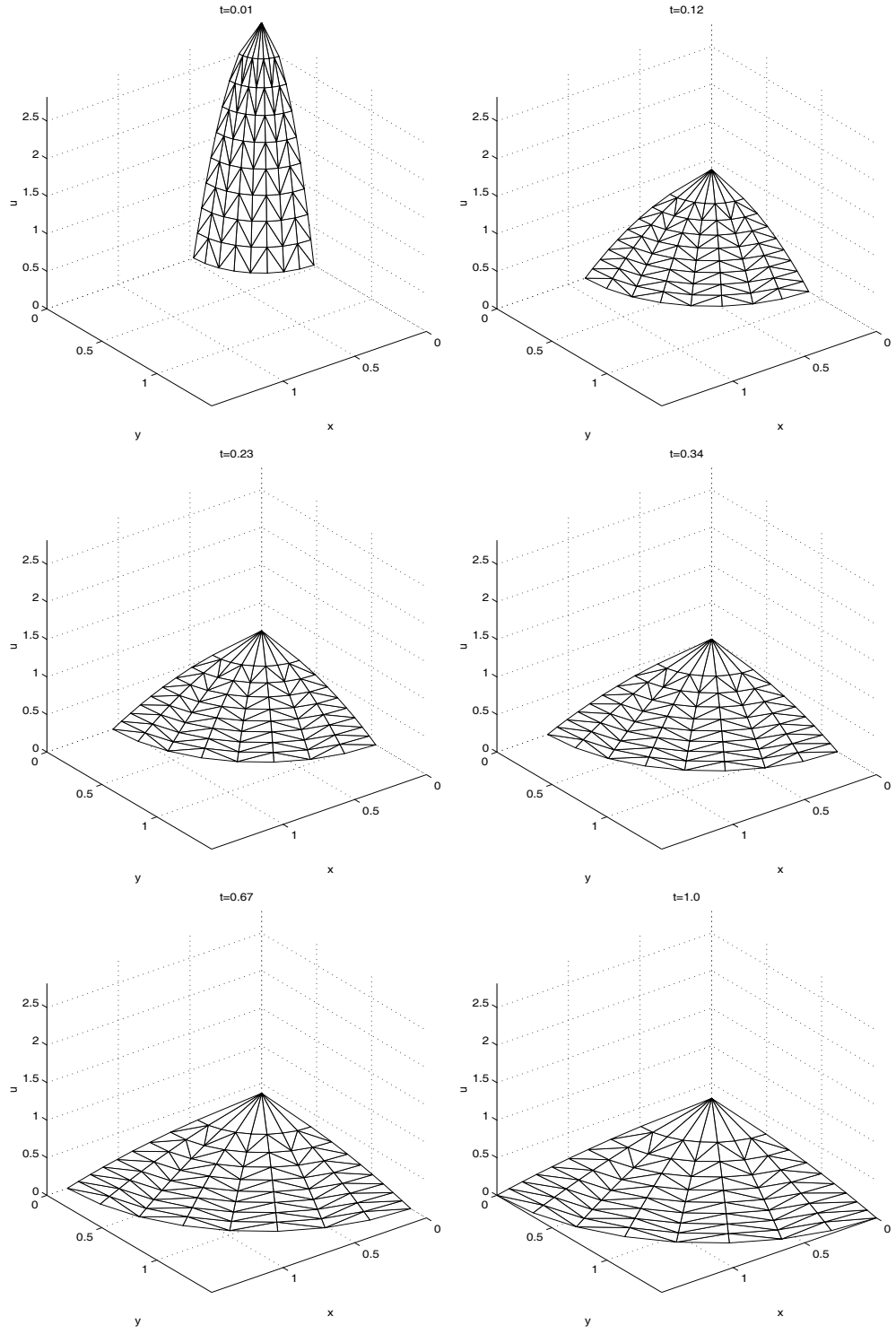

Figure 7.17: Approximate solutions ($m = 1$) using a 'patchwise' approach in conjunction with an exact form of mesh speed. Calculations using a tolerance = 10^{-12} on mesh 2.

of the BDF routine you can clearly see the solution becoming wildly unstable, the probable cause of the halting of the NAG routine. Figure 7.19 shows the patchwise solution process being used on a mesh with the same construction as the mesh used in the previous example, only with m, the power in the diffusion coefficient equal to 2. Again the NAG routine fails, this time at $t \approx 0.042913$, the plots demonstrating the same behaviour. Both these example suggest that the instabilities suspected in the method lie primarily in the solution of u from the mesh. These undesirable effects are present when using the exact form of the node speeds, which is significant since in this form the velocities are derived entirely from the current solution with no least squares solution for the node speeds involved. This does not prove however that such effects are not entirely present in the calculations of the node speeds in the general case. The next section attempts to illustrate why the methodology seems to be successful in one dimension and does not translate as easily to a higher-dimensional problem.

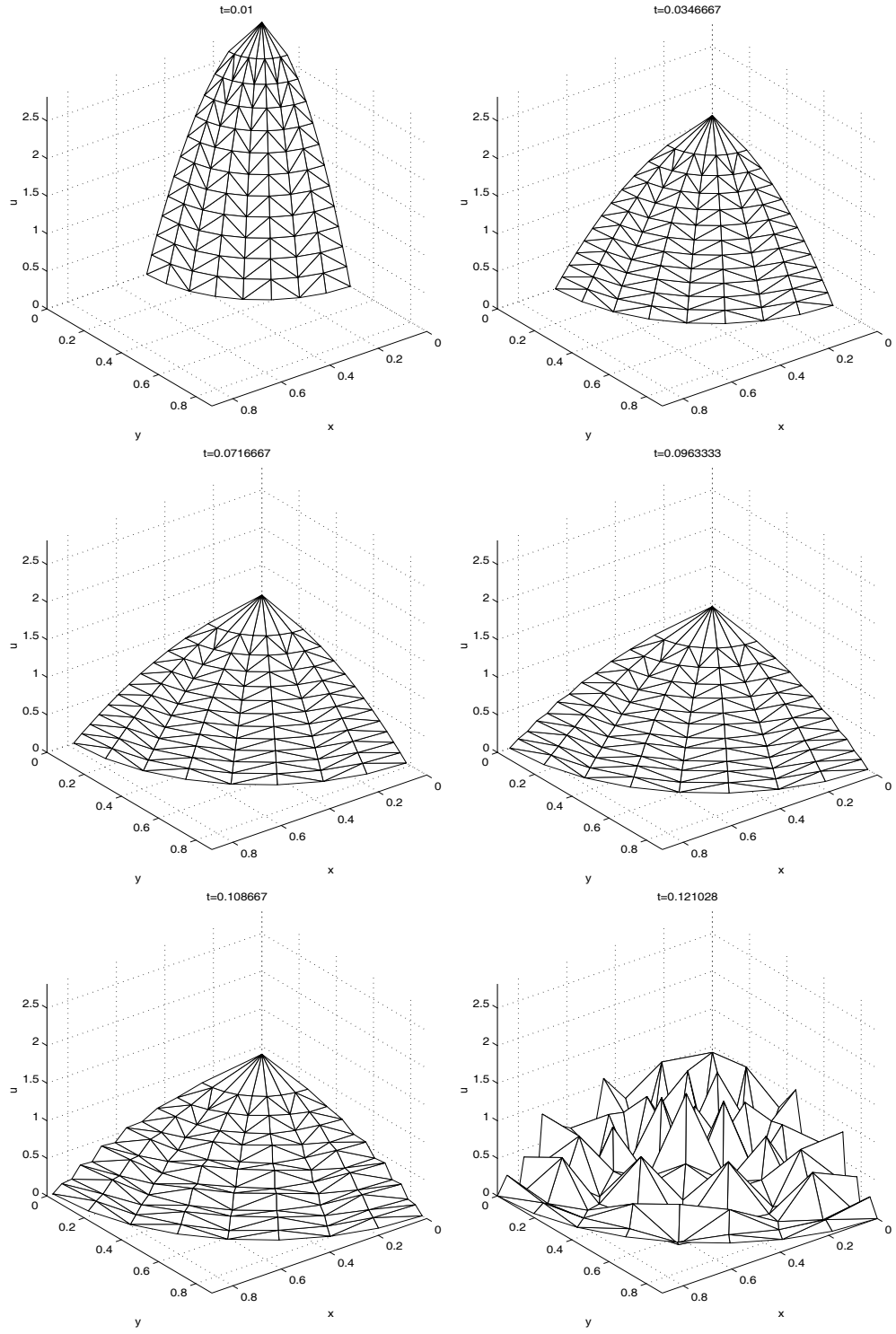

Figure 7.18: SVD solution until BDF failure in integration on modified mesh 2 with $m = 1$.

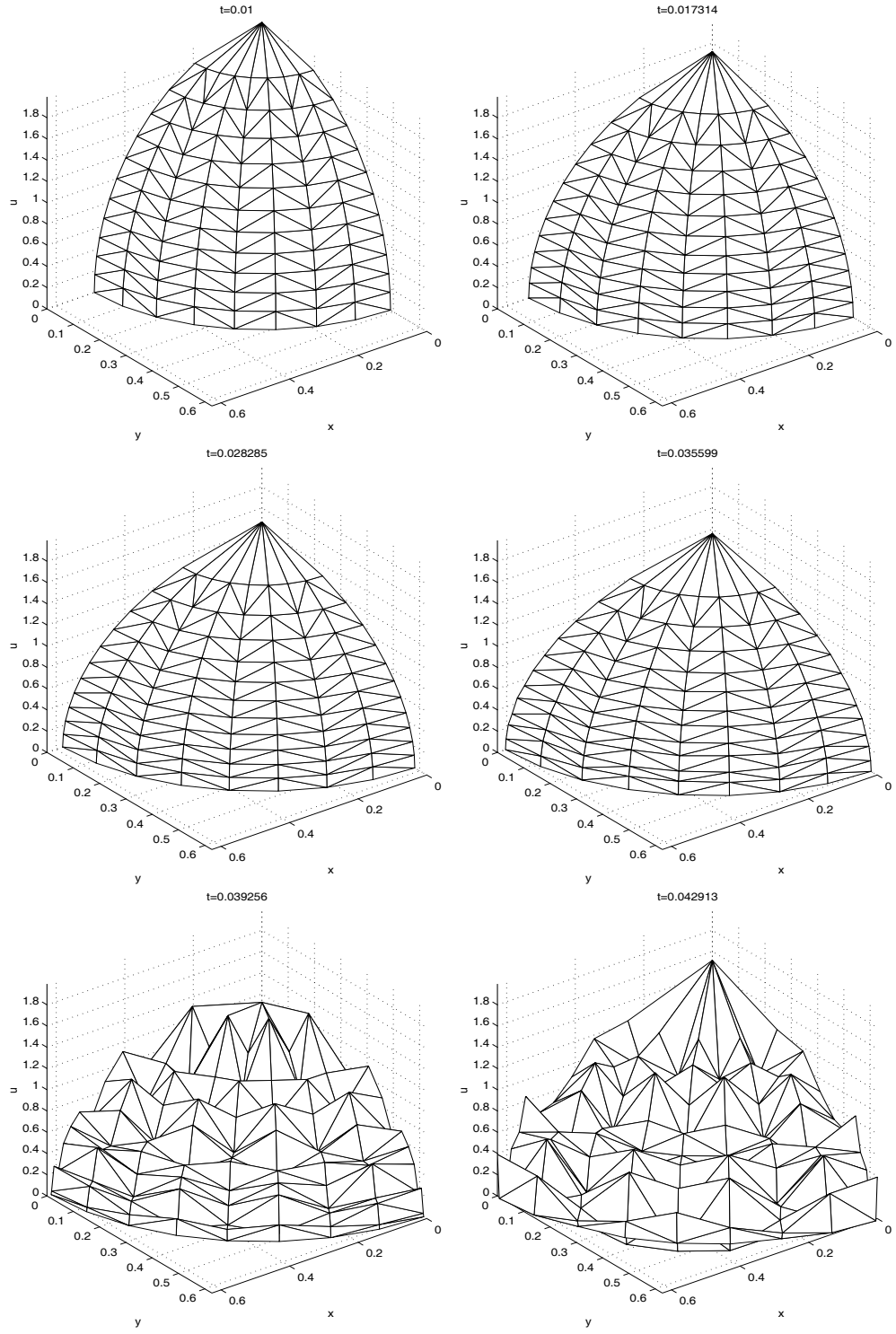

Figure 7.19: Patchwise solution until BDF failure in integration on modified mesh 2 with $m = 2$

7.3 A Brief Analysis

The previous section concluded that instabilities, primarily in the solution of u from the correct mesh, made the resulting moving mesh method in two dimensions extremely difficult to solve. However, we have seen that in one dimension the method provided accurate and, most notably, reliable results for varying mesh resolutions and strict error tolerances when integrating forward in time. We now provide a brief analysis as to why these difficulties have been encountered in two dimensions.

We are considering movement via mass conservation, so over an arbitrary computational element or cell Ω with boundary $\partial\Omega$ we have that

$$\frac{d}{dt}\int_{\Omega} u d\Omega = \int_{\Omega} u_t + \oint_{\partial\Omega} u\underline{\dot{x}}.\underline{\hat{n}} dS = 0.$$

Given a PDE of the form

$$u_t = \nabla.\underline{F}$$

where $\underline{F}$ is some flux function, our conservation law becomes

$$\oint_{\partial\Omega} (u\underline{\dot{x}} + \underline{F}).\underline{\hat{n}} dS = 0. \tag{7.25}$$

We aim to couple the resulting moving mesh relation to a discrete quadrature rule, prescribing, θ the mass conserved in the cell, i.e.

$$\int_{\Omega} u d\Omega = \theta. \tag{7.26}$$

Using piecewise linear approximations to u and $\underline{\dot{x}}$, equations (7.25) and (7.26), when applied over the complete computational domain, generate two systems respectively of the form

$$A(\underline{u})\underline{\dot{x}} = \underline{f}(\underline{u}, \underline{x}) \tag{7.27}$$

and

$$B(\underline{x})\underline{u} = \underline{\theta}. \tag{7.28}$$

In one dimension, this pairing works well, since both systems are square and invertible. Moreover, it can be shown that $B(\underline{x})$ is diagonally dominant and well-conditioned. Hence

$$\underline{u} = B(\underline{x})^{-1}\underline{\theta}$$

giving

$$\begin{aligned} \dot{\underline{x}} &= A(\underline{u})^{-1}\underline{f}(\underline{u}), \underline{x}), \\ &= A(B(\underline{x})^{-1}\underline{\theta})^{-1}\underline{f}(B(\underline{x})^{-1}\underline{\theta}), \underline{x}), \end{aligned} \tag{7.29}$$

which gives a well-defined ODE system for $\dot{\underline{x}}$.

However in two dimensions, both systems (7.27) and (7.28) are, in general, non-square. Employing some suitable matrix solver yields a least squares solution $\bar{\underline{u}}$, but it is at this step that errors are introduced, since $\bar{\underline{u}}$ does not satisfy (7.28) exactly. This in turn has widespread implications for the solution of (7.27), since the conservation law is violated by the least squares solution $\bar{\underline{u}}$.

In one dimension the structure of the computational mesh allows the discrete masses $\underline{\theta}$ to be conserved exactly, allowing the moving mesh equations to remain valid. Hence, by ensuring the discrete conservation of mass, we preserve the validity of the moving mesh equations. It is worth noting that in two dimensions when using the patchwise solution approach to recover the solution u from the mesh, the matrix system to solve is actually square, and yet instabilities still appear in the solution process. However, further analysis of these matrices reveal a general trend for them to be badly conditioned. The left hand side of Figure 7.20 shows the condition of the patchwise system matrix generated for initial conditions on meshes of successively greater resolution with $m = 1$. The remaining plot illustrates the effect of increasing the power m of the diffusion coefficient in these initial conditions on meshes with a set number of nodes. Both clearly show that as the mesh contains more points, or for increasing m, the matrix becomes more ill-conditioned, more significantly in the latter case. The coefficients of the matrix in the patchwise approach consist of areas of cells, hence in the case of increasing m, the gradient of u grows and as a result of

our meshing strategy, the increased resolution creates smaller and smaller triangles. Hence the ratios of the coefficients of the matrix A increase dramatically. Since the coefficients of the non-square SVD matrix are all equal to 1 the conditioning does not seem to increase as dramatically, and remains unchanged irrespective of m since it is determined primarily by the connectivity of the grid. However, as noted before this approach cannot guarantee a unique solution.

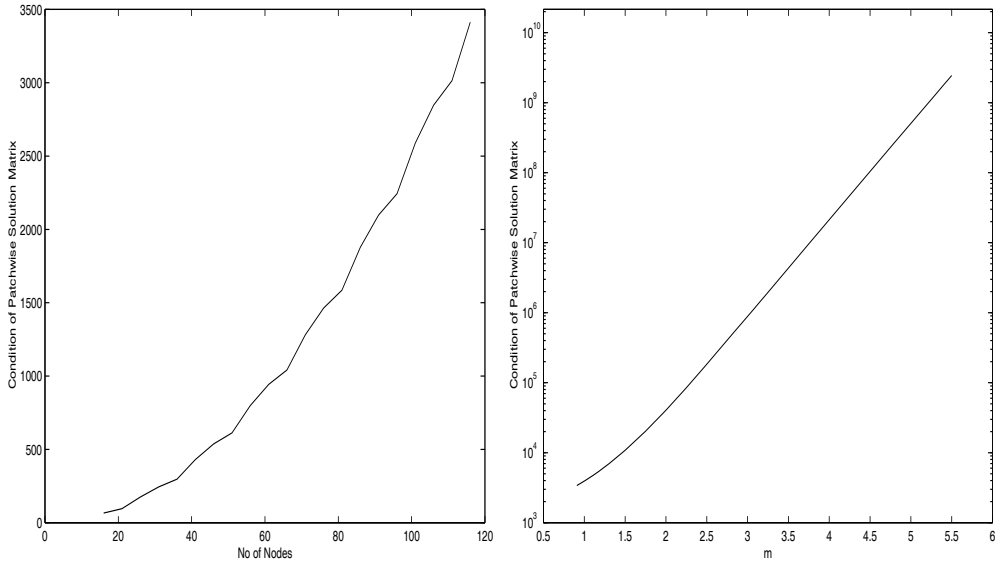

Figure 7.20: Conditioning of Patchwise Solution Matrix with increasing number of nodes (Left) and increasing m (Right).

7.5.1 Improving Conditioning

Having ascertained that both the SVD and Patchwise approaches to u solution are both badly conditioned, we finally attempt to improve the conditioning of the patchwise solution method. We stated above that a primary factor behind the limited success in implementing the ideas which performed well in one dimension into two dimensions, was the ill-conditioned non-square systems involved in the higher dimensional algorithm. Noting that the patchwise system is such a square system, we attempt to improve the conditioning, but relaxing the constraints of conserved mass over an entire patch.

The structure of our grids means that by considering only a 'half-patch' in the

upwind radial direction we form a better conditioned matrix, to be precise a collection of tridiagonal relations between contours. Figure 7.21 shows a section of grid with a patch associated with node i lying on contour c_2 and between two contours c_1 and c_3 with associated u values such that $u_3 > u_2 > u1$. Our aim is to sum the equations representing the conserved discrete mass (7.10) over the shaded region only.

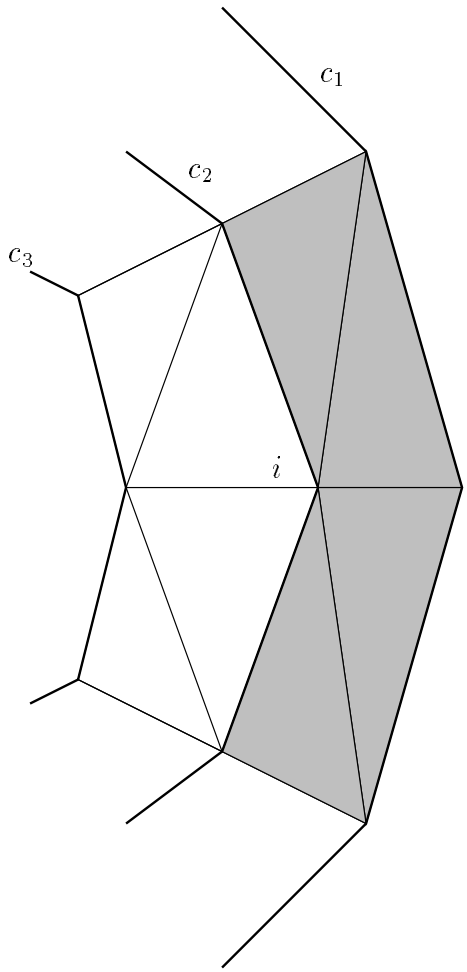

Figure 7.21: The half patch associated with node i

In a sense this very much falls in line with the approach in one dimension, this time tracing backwards in the radial direction, constructing the solution over a contour instead of a single point.

Figure 7.22 shows the conditioning for both the half-patch and full patch approaches on mesh of increasing resolution. It is clear to see the improvement in

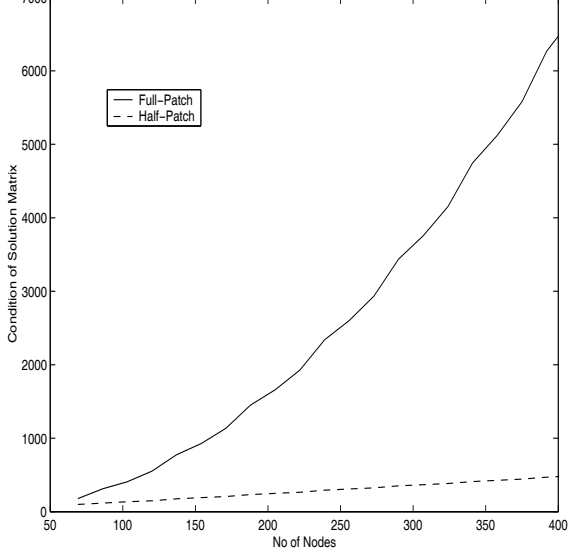

Figure 7.22: Conditioning of full and half patchwise approaches on meshes of increasing resolution.

the conditioning of this half-patch approach. However, despite this improvement, the method still exhibits instabilities. These are illustrated using the exact same strategy of using consistent grid and solution parameters as for the results produced in Figure 7.18. Figure 7.23 shows the intermediate plots as the solution reaches the time of failure reached by the BDF routine($t \approx 0.0169277$). Interestingly enough, the instabilities in this case seem to centre around the maximum, whereas the full patch solution gave such behaviour throughout the mesh (see Figure 7.18). In hindsight this is logical, since at the origin there is no distinction as between a full and half-patch, due to our use of symmetry. Oscillations exist in the outer regions of the solution, but they manifest themselves more prominently at this point.

7.6 Summary

This chapter has been focused on the aim of translating the moving mesh method developed in one dimension in Chapter 5 to higher dimensions. Following the development of the method in one dimension, we began by attempting the radially symmetric PME first using a radial form of the mass monitor in one-dimension and then using the mass monitor in two dimensions on a suitably refined triangular mesh.

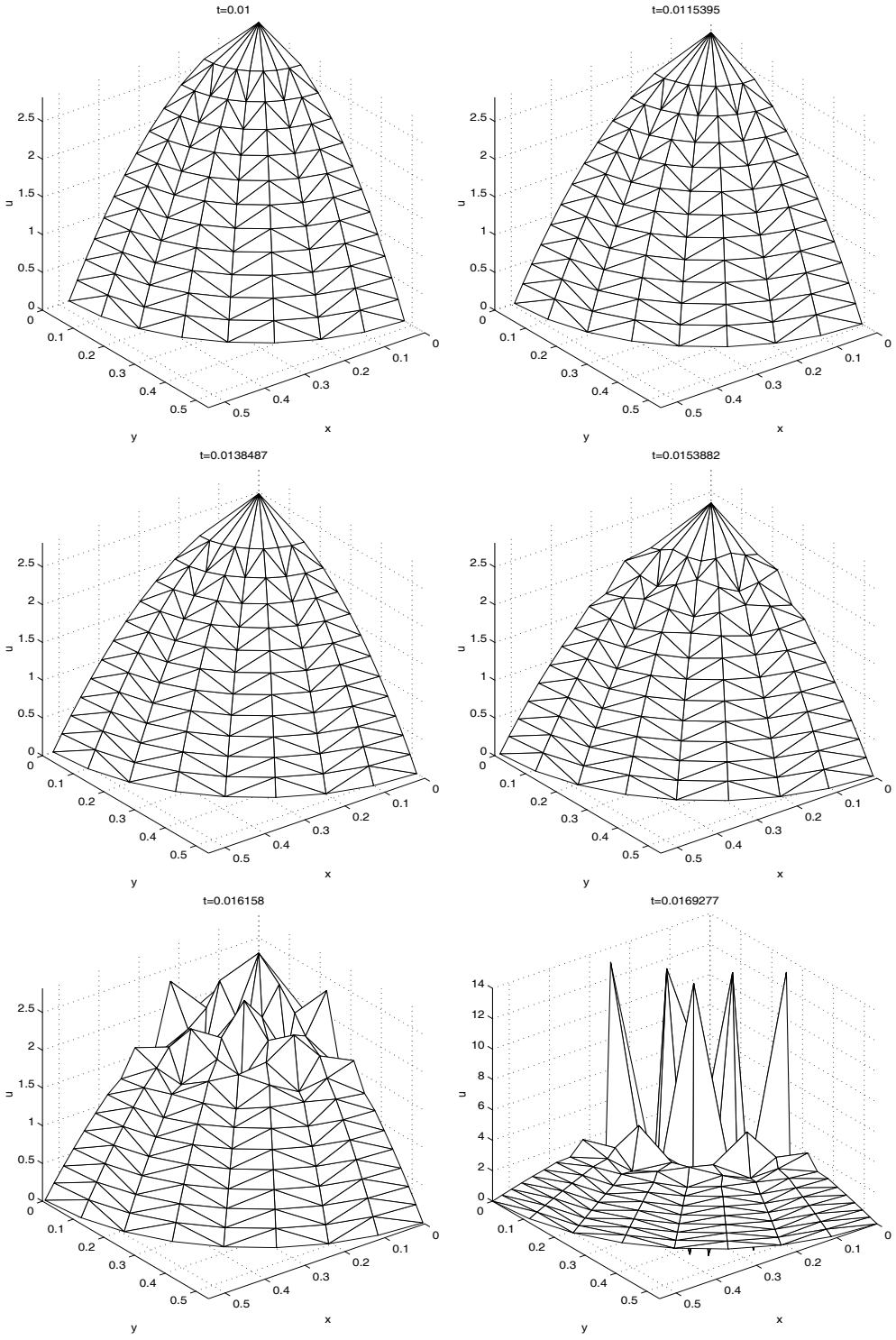

Figure 7.23: Half-Patch solution until BDF failure in integration on modified mesh 2 with $m = 1$

Initial results proved satisfactory for simple meshes using low error tolerance in our ODE package. However, we would ideally require a more robust performance allowing us to then continue to solve problems for greater powers of m.

Our method involves the solution of two overdetermined systems. Using SVD techniques we can guarantee the existence of a solution, but this will almost certainly not ensure all equations concerned will be satisfied. By taking advantage of the geometry of the radial case of the PME, we eliminated the system relating to the nodal velocities. This highlights the problems involved in constructing a stable solution from the discrete static conservation law over each cell (7.10). In all, four such approaches were attempted, all of which exhibited unstable behaviour in u as the grid evolved.

Due to the nature of the overdetermined system relating conserved discrete mass, the solution u and the grid x, none of the methods can guarantee that all equations are satisfied. This fact, in conjunction with the strict boundary conditions imposed on u at the moving boundary, may be the source of unstable behaviour exhibited. Other contributory factors may include the lack of a sustained grid resolution along contours as the boundary expands and the distance between nodes in the direction tangential to the contours increases. Moreover the linear quadrature technique used for approximating the quantities of mass conserved in each cell may be too weak and extra degrees of freedom may be needed to allow the grid to move. Whatever, the cause, we have outlined a brief and basic analysis as to how the lack of square and invertible systems relating grid speeds and solution to the current state of the mesh prohibits the direct implementation of our previous work to higher dimensions.

Chapter 8

Conclusions and Further Work

Grid adaption and the use of moving meshes has evolved dramatically over recent years, becoming an essential tool in the successful numerical solution to a wide variety of applications. The ability of a mesh to automatically adjust its distribution in order to resolve steep or sharp solution variations can aid the numerical analyst in gaining effective control over computational resources.

This thesis has illustrated two contrasting moving mesh methods for the solution of parabolic PDEs. To be more precise, deficiencies found when computing numerical solutions using an initial static method have motivated the development of a dynamic approach which was able to resolve difficulties found in the application of the former algorithm. This final chapter serves as a summary of the work presented and suggests possible future avenues of study.

We began by recalling the different approaches used in the construction of a moving mesh. Chapter 1 introduced the three styles of grid adaption methods and the motivation for the choice of employing a moving mesh algorithm. We then described how some methods are considered to be static, with a redistribution of nodes undertaken at intermittent times throughout the solution, and how some were in effect dynamic, with a separate equation or relation prescribing a continuous strategic movement of mesh points.

In Chapter 2 we gave an overview of existing grid generation and moving mesh techniques. Of main concern were those which constitute an r-refinement approach. These methods relocate nodes to areas of interest without increasing the arithmetic

overheads caused by adding extra points into the grid. We began by introducing grid generation techniques and in particular the equidistribution and functional approaches. Our review then proceeded to explain how these existing ideas were adapted to include velocities of mesh points such that, for time dependent problems, the mesh adapts with the evolution of the solution to provide suitable resolutions around troublesome regions. Also introduced was the idea of self-similar solutions and recent work which suggests that specific mesh methods should be employed such that the numerical approximations inherit theoretical properties of the underlying p.d.e. This was illustrated with particular reference to the PME, which admits an analytical solution, used later for qualitative assessment of the mesh methods covered.

Chapters 3 and 4 introduced a static regridding technique called *Contour Zoning*. In one dimension this is a standard finite-volume method, the grid movement being driven by the preservation of an easily enforceable equidistribution rule using the gradient monitor. Moreover, in two dimensions this form of monitor function is well suited to solving for heights of contours comprising a number of individual mesh points. This solution technique obviously reduces the computational overheads. The method was used to generate solutions to the PME, but encountered difficulties in a semiconductor modelling problem. In one dimension the problem was found to be unable to conserve mass, as required of the solution. This problem results from the regridding procedure when solutions are interpolated between changing states of the grid. This situation could be improved by using a more refined redistribution process but not completely eradicated.

In the case of the porous media problem, which involves a moving boundary, the Contour Zoning method was found to be unsuitable since for an accurate representation of this feature the speed of the moving front was required, which was judged to be inconsistent with the regridding strategy. Upon moving to two dimensions the same problems persisted. Moreover, when using the Contour Zoning technique it was found that the method would prohibit changes in geometry of the contours since the solution approach crudely averages flux approximations around the contour and in some sense generalises the resulting effects. It became obvious that, with the

PME in particular, a more dynamic approach would be needed in order to preserve mass, accurately deal with the moving boundary and later, in two dimensions, allow the freedom of movement for individual nodes.

In response to the problems experienced with the Contour Zoning technique, we developed in Chapter 5 a moving mesh algorithm with direct reference to the PME. The technique begins with an equidistribution principle concerned with equally distributing mass within computational cells. Taking advantage of some of the properties of the PME, the problem is solved entirely in terms of the grid positions. The resulting solution can then be reconstructed over the mesh using an algebraic trapezium rule approximation to the mass contained within each cell. Moreover the method can be thought of purely as mesh movement through mass conservation rather than through equally distributing mass, and hence the method could be modified slightly such that steep moving boundaries were automatically resolved by placing smaller quantities of mass in regions of high solution variation. In addition, the moving boundary is incorporated neatly into the framework of the method.

In Chapter 5 we extended the method further to the use of equidistribution via more complex monitor functions. First a gradient monitor, as used for the Contour Zoning method, was used, following the same solution ideas as previously using in conjunction with the mass monitor. After limited success, a combination of the two functions was devised which moves the mesh automatically into an efficient distribution without the need for the prior arrangement of scaled quantities of mass.

In Chapter 6 we applied this new approach to problems which do not possess those properties naturally advantageous to the method, as in the PME. The PME has a known solution value throughout time, namely $u = 0$ at the moving boundary, and satisfies conservation of mass. In the previous chapter the known solution value allowed us to easily impose the algebraic relation to reconstruct our solution from the current states of the mesh. For the semi-conductor problem we do not have this luxury, Neumann conditions imposed at both boundaries denying us such a condition. However we were able to couple the ODE system resulting from the mesh movement to a local solution procedure at the relevant boundary. We were then able to proceed as before using another style of combination monitor. The

second property of the PME, which proved critical in the derivation of the original form of the method, was mass conservation. For an inhomogeneous problem, involving the blow-up of the solution around a single point, we had to allow for the rapid accumulation of mass over the whole domain due to a non-linear source term powering the combustion of the solution around the origin. In a similar way as when tackling the semi-conductor problem, an expression for the rate of gain of total mass was coupled to the grid movement system. The resulting solution was verified when it produced a favourable approximation of the time of blow-up when compared to previous numerical solutions existing in the literature.

Finally, in Chapter 7 we attempted to interpret the existing mass conservation grid movement idea directly into two dimensions. In terms of derivation a direct analogy can be found from the work in one-dimension. However, problems were encountered since for the grids considered both the ODE system for the node velocities and the algebraic mass conservation equations produced non-square systems, which led to the necessity for averaging and generated instabilities in the solution. Again, using the PME as a focus of our attention, it was found that the construction of a least squares solution for the solution u in terms of conserved mass was ill-conditioned. In all, four approaches were constructed for the solution of the system, all exhibiting unstable solution behaviour when solved on a mesh with any kind of adequate resolution or with a reasonable error tolerance for the integration in time. It was concluded that the one-dimensional approach which allows for square well-conditioned sets of equations to be solved does not readily generalise to higher dimensions without a new idea. In theory the methodology extends to higher dimensions but in practice an adequate solution to these problems has yet to be found.

8.1 Further Work

To conclude, we now suggest possible areas of further research as a result of our findings.

The most immediate area of interest is the search for a reliable solution technique

for the moving mesh algorithm in two dimensions. The main objective would be to either construct the mesh to allow for a square system of equations for both the nodal velocities $\underline{\dot{x}}$ and the solution u, or to deduce a reliable and more importantly stable method for finding a solution which will approximately satisfy all equations without creating escalating oscillations.

Despite the limited success of the moving mesh method to generate solutions in two dimensions, this thesis has presented an interesting solution technique for problems in one dimension. It is obvious, though, that the method still needs further work and application to other types of problem to test its robustness and suitability for widespread application. For example, could we apply the technique to hyperbolic problems? The success of the mass monitor in both one-dimensional and radial coordinate cases suggest that a similar approach could possibly be implemented for the solution of hyperbolic conservation laws.

It has also become apparent that the one-dimensional work could be implemented using the arc-length monitor.

$$M(u) = \sqrt{1 + u_x^2}$$

A discrete approximation to the equidistribution quantity $\theta(t)$ would in this case take the form

$$1 + u_x^2 = \theta(t)^2 \approx 1 + \left(\frac{u_{i+1} - u_i}{x_{i+1} - x_i} \right)^2 .$$

which could be rewritten as

$$(x_{i+1} - x_i)\sqrt{(\theta(t)^2 - 1)} = u_{i+1} - u_i .$$

In the case of the PME, this would present the required square, invertible system for the construction of u over the current grid. Although our combination monitor (5.21) does in some sense mimic the arc-length monitor with a suitable choice of α, it would be nice to eliminate the need for a user-defined parameter.

The final extension we wish to comment on is motivated by the use of a refined grid in conjunction with the mass monitor in Section 5.2. Here we strategically

placed smaller amounts of mass in the region behind the evolving steep moving front to resolve the large variations in u. Budd has suggested that this could be considered as a 'skew' distribution of mesh points and could be formalised as a variation of the equidistribution principle. Taking the form (2.4) from Chapter 2, this could be written as

$$\int_{x(\xi_i)}^{x(\xi_{i+1})} M d\tilde{x} = \zeta(\xi) \int_0^1 M d\tilde{x}$$

where $\zeta(\xi)$ is some weighting function which will influence the distribution of M over the mesh automatically. The style in which we refined our mesh in Section 5.2 would imply that $\zeta(\xi)$ would be a decreasing exponential function. Obviously for a practical application, some prior knowledge would be needed of the solution behaviour, but it does provide an interesting abstraction of the equidistribution principle.

References

[1] S. Adjered and T.E. Flaherty. A moving finite element method with error-estimation and refinement for one-dimensional time dependent partial differential equations. *SIAM Journal on Numerical Analysis*, 23:778–796, 1986.

[2] D.A. Anderson. Adaptive mesh schemes based on grid speeds. *A.I.A.A*, 1:311, 1983.

[3] M.J. Baines. Grid adaption via node movement. *Applied Numerical Mathematics*, 26:77–96, 1998.

[4] M.J. Baines. Least squares and approximate equidistribution in multi-dimensions. *Numerical Methods Partial Differential Equations*, 15:605–619, 1999.

[5] G.I. Barenblatt. In *Similarity, self-similarity, and intermediate asymptotics*. New York : Consultants Bureau, 1979.

[6] G. Beckett, J.A. Mackenzie, J. Ramage, and D.M. Sloan. On the numerical solution of one-dimensional partial differential equations using adaptive methods based on equidistribution. Technical Report 2000/02, Strathclyde Mathematics Research Report, 2000.

[7] G. Beckett, J.A. Mackenzie, and M.L. Robertson. A moving mesh finite element method for the solution of two-dimensional stefan problems. Technical Report 99/26, Dept of Mathematics, University of Strathclyde, 1999.

[8] K.W. Blake. Contour zoning. Technical Report September 1998, MSc Dissertation, Department of Mathematics, University of Reading, 1998.

[9] K.W. Blake. New developments in contour zoning. Technical Report May 1999, Numerical Analysis Report, Department of Mathematics, University of Reading, 1999.

[10] K.W. Blake. The use of moving grids in contour zoning. Technical Report 8/99, Numerical Analysis Report, Department of Mathematics, University of Reading, 1999.

[11] J.G. Blom, J.M. Sanz-Serna, and J.G. Verwer. On simple moving grid methods for one-dimensional evolutionary partial differential equations. *Journal of Computational Physics*, 74:191–213, 1988.

[12] J.U. Brackbill. An adaptive grid with directional control. *Journal of Computational Physics*, 108:38–50, 1993.

[13] J.U. Brackbill and J.S. Saltzman. Adaptive zoning for singular problems in two dimensions. *Journal of Computational Physics*, 46:342–36, 1982.

[14] C.J. Budd, S. Chen, and R.D. Russell. New self-similar solutions of the nonlinear schrodinger equation with moving mesh computations. *Journal of Computational Physics*, 152:756–789, 1999.

[15] C.J. Budd and G. Collins. An invariant moving mesh scheme for the nonlinear diffusion equation. Technical Report 19/08/96, School of Mathematics, University of Bath, 1996.

[16] C.J. Budd, G.J. Collins, W. Huang, and R.D Russell. Self-similar numerical solutions of the porous-medium equation using moving mesh methods. *Phil. Trans R. Soc. London*, 357:1047–1077, 1999.

[17] C.J. Budd, W. Huang, and R.D. Russell. Moving mesh methods for problems with blow-up. *SIAM Journal of Scientific and Statistical Computation*, 1996.

[18] C.J. Budd and M.D. Piggott. The geometric integration of scale-invariant ordinary and partial differential equations. *Journal of Computational and Applied Mathematics*, 128:399–422, 2001.

[19] W. Cao, W. Huang, and R.D. Russell. An r-adaptive finite element method based upon moving mesh partial differential equations. *Journal of Computational Physics*, 149:221–244, 1999.

[20] W. Cao, W. Huang, and R.D. Russell. A study of monitor functions for two dimensional adaptive mesh generation. *SIAM Journal of Scientific and Statistical Computations*, 20:1978–1994, 1999.

[21] G.F Carey and H.T. Dinh. Grading functions and mesh redistribution. *SIAM Journal on Numerical Analyses*, 22:1028–1040, 1985.

[22] D Catherall. The adaption of structured grids to numerical solutions for transonic flow. *Journal of Numerical Methods in Engineering*, 32:921–937, 1991.

[23] C De Boor. In *Good Approximation by Splines with Variable Knots II,Springer Lecture Notes Series 363*. Springer-Verlag, Berlin, 1973.

[24] I. Demirdžić and M. Perić. Finite volume method for prediction of fluid flow in arbitrarily shaped domains with moving boundaries. *International Journal for Numerical Methods in Fluids*, 10:771–790, 1988.

[25] I. Demirdžić and M. Perić. Space conservation law in finite volume calculations of fluid flow. *International Journal for Numerical Methods in Fluids*, 8:1037–1050, 1988.

[26] V.E. Denny and R.B. Landis. A new method for solving two-point boundary value problems using optimal node distribution. *Journal of Computational Physics*, 9:120–137, 1972.

[27] E.A. Dorfi and L.O.C. Drury. Simple adaptive grids for 1-d initial value problems. *Journal of Computational Physics*, 69:175–195, 1987.

[28] L. Dresner. In *Similarity Solutions of Non-linear Partial Differential Equations*, volume 88, pages 349–388. Pitman Research Notes in Mathematics Series, London:Longman, 1983.

[29] D.G. Dritschel. Introduction to contour dynamics for the euler equations in two dimensions. *Journal of Computational Physics*, 135:217–219, 1997.

[30] A.S. Dvinsky. Adaptive grid generation from harmonic maps on riemannian manifolds. *Journal of Computational Physics*, 95:450–476, 1991.

[31] H.A. Dwyer. Grid adaption for problems in fluid dynamics. *A.I.A.A*, 22:1705–1712, 1984.

[32] C.L. Farmer. An application of triangulations to the building of structured grids. *Paper for presentation at the 6th European Conference on the Mathematics of Oil Recovery.*, 1998.

[33] J.E. Flaherty, J.M. Coyle, and R Ludwig. On the stability of mesh equidistribution strategies for time-dependent partial differential equations. *Journal of Computational Physics*, 62:26–39, 1986.

[34] R.M. Furzeland, J.G. Verwer, and P.A. Zegeling. A numerical study of three moving-grid methods for one-dimensional partial differential equations which are based on the method of lines. *Journal of Computational Physics*, 89:349–388, 1990.

[35] G.H. Golub and C.F. Van Loan. In *Matrix Computations*, chapter 5. The John Hopkins University Press, 1996.

[36] Numerical Algorithms Group. The NAG fortran library manual. In *http://www.nag.co.uk/numeric/fl/manual/html/FLlibrarymanual.asp*. NAG Ltd, 2001.

[37] M.J. Halsall and H.H. Inston. Contour-zoning: A new neutronics method incorporated into the prototype code mytmus. *Nuclear Energy*, 28:351–352, 1989.

[38] M.J. Halsall and H.H. Inston. A prototype neutron-kinetics code mytmus incorporating a new method - contour-zoning. *PHYSOR Conference held in Marseilles, April 1990*, 1990.

[39] W. Haung. Practical aspects of formulation and solution of moving mesh partial differential equations. Technical Report 99-11-02, University of Kansas, 1999.

[40] D.F. Hawken, J.J. Gottlieb, and J.S. Hansen. Review of some adaptive node-movement techniques in finite -element and finite difference solutions of partial differential equations. *Journal of Computational Physics*, 95:254–302, 1991.

[41] R.G. Hindman and J Spencer. A new approach to truly adaptive grid generation. *A.I.A.A*, 1, 1983.

[42] J.M. Hobbs. In *A Moving Finite Element Approach for Semiconductor Process Modelling in 1-D.* MSc Dissertation, Department of Mathematics, University of Reading, 1993.

[43] W. Huang, Y. Ren, and R.D. Russell. Moving mesh methods based on moving mesh partial differential equations. *Journal of Computational Physics*, 113:279–290, 1994.

[44] W. Huang, Y. Ren, and R.D. Russell. Moving mesh partial differential equations (MMPDEs) based on the equidistribution principle. *SIAM Journal on Numerical Analysis*, 31:709–730, 1994.

[45] W. Huang and R.D. Russell. Analysis of moving mesh partial differential equations with spatial smoothing. *SIAM Journal on Numerical Analysis*, 34:1106–1126, 1997.

[46] W. Huang and R.D. Russell. A high-dimensional moving mesh strategy. *Applied Numerical Mathematics*, 26:63–76, 1997.

[47] W. Huang and R.D. Russell. Moving mesh strategy based on a gradient flow equation for two-dimensional problems. *SIAM Journal of Scientific Computing*, 20:998–1015, 1999.

[48] W. Huang and R.D. Russell. Moving mesh strategy based upon a heat flow equation for two dimensional problems. *SIAM Journal of Scientific Computing*, 20:998–1015, 1999.

[49] W. Huang and R.D. Russell. Adaptive mesh movement - the MMPDE approach and its applications. *Journal of Computational and Applied Mathematics*, 128:383–398, 2001.

[50] W. Huang and D Sloan. A simple adaptive grid method in two-dimensions. *SIAM Journal of Scientific Computing*, 4:776–797, 1994.

[51] M.E. Hubbard. In *Multidimensional Upwinding and Grid Adaption for Conservation Laws*. Ph.D. Thesis, University of Reading, Department of Mathematics, 1996.

[52] H.H. Inston. Contour zoning. 1998.

[53] K. King and C.P. Please. Diffusion of dopant in crystalline solicon - an asymptotic analysis. *I.M.A.Journal of Applied Mathematics*, 1986.

[54] P. Knupp. Mesh generation using vector fields. *Journal of Computational Physics*, 119:142–148, 1995.

[55] P. Knupp and S. Steinberg. In *Fundamentals of Grid Generation.*, chapter 3, page 82. CRC Press, 1994.

[56] B. Larrouturou. A conservative adaptive method for flame propagation. *SIAM Journal of Scientific and Statistical Computation*, 10:742–755, 1989.

[57] S.J. Leary. In *Least-Squares Methods with Adjustable Nodes for Steady Hyperbolic PDEs*. Ph.D. Thesis, University of Reading, Department of Mathematics, 1999.

[58] S. Li, L. Petzold, and Y Ren. Stability of moving mesh systems of partial differential equations. *SIAM Journal of Scientific Computing*, 20:719–738, 1996.

[59] J.A. Mackenzie. Moving mesh finite volume methods for one-dimensional evolutionary partial differential equations. Technical Report 96/02, Dept of Mathematics, University of Strathclyde, 1996.

[60] J.A. Mackenzie and M.L. Robertson. A moving mesh method for the solution of the one-dimensional phase-field equations. Technical report, Dept of Mathematics, University of Strathclyde, 2000.

[61] A.J. Malcolm. In *Data Dependent Grid Generation*. Ph.D. Thesis, University of Reading, Department of Mathematics, 1991.

[62] K.W. Morton. In *Numerical Solution of Convection-Diffusion Problems*, chapter 3, page 82. Chapman & Hall, 1996.

[63] L.S. Mulholland, Y. Qiu, and D.M. Sloan. Solution of evolutionary partial differential equations using adaptive finite differences with pseudospectral post-processing. *Journal of Computational Physics*, 131:280–298, 1997.

[64] J.D. Murray. In *Mathematical Biology*, pages 349–388. New York: Springer, 1989.

[65] L. Petzold. Observations on an adaptive moving grid method for one-dimensional systems of partial differential equations. *Applied Numerical Mathematics*, 3:347–360, 1987.

[66] C.P. Please and P.K. Sweby. A transformation to assist numerical solution of diffusion equations. Technical Report 5/86, Numerical Analysis Report, University of Reading, 1986.

[67] Y. Qiu and D.M. Sloan. Numerical solution of the fisher's equation using a moving-mesh equation. Technical Report 1997/22, Strathclyde Mathematics Research Report, 1997.

[68] Y Ren and R.D. Russell. Moving mesh techniques based upon equidistribution and their stability. *SIAM Journal of Scientific and Statistical Computing*, 13:1265–1286, 1992.

[69] J.M. Stockie, J.A. Mackenzie, and R.D. Russell. A moving-mesh method for one-dimensional hyperbolic conservation laws. Technical Report 2000/06, Dept of Mathematics, University of Strathclyde, 1999.

[70] J.F. Thompson, F.C. Thames, and C.W. Mastin. Automatic numerical generation of body-fitted curvi-linear coordinate system for field containing any number of arbitrary two-dimensional bodies. *Journal of Computational Physics*, 15:299–319, 1995.

[71] J. Thuburn. Private communication. *Department of Meteorology, University of Reading*, 1998.

[72] J.G. Verwer, J.G. Blom, R.M. Furzeland, and P.A. Zegeling. In *Adaptive Methods for Partial Differential Equations*, page 160. Society for Industrial and Applied Mathematics, 1989.

[73] D.S. Watkins. In *The Fundamentals of Matrix Computations*, chapter 7. John Wiley & Sons Inc, 1991.

[74] A.B. White. On selection of equidistributing meshes for two-point boundary-value problems. *SIAM Journal on Numerical Analysis*, 16:472–502, 1979.

[75] A.B. White. On the numerical solution of initial/boundary-value problems in one-space dimension. *SIAM Journal on Numerical Analysis*, 4:683–697, 1987.

[76] J. Williams. Private communication. *Department of Mathematics, University of Bath*, 2001.

[77] A.M. Winslow. Numerical solution of the quasi-linear poisson equation in a non-uniform triangle mesh. *Journal of Computational Physics*, 1:128–138, 1967.