THE UNIVERSITY OF READING

Department of Mathematics and Statistics

Modelling time-dependent partial differential equations using a moving mesh approach based on conservation

Tamsin E. Lee

November 2011

Thesis submitted for the degree of Doctor of Philosophy

Abstract

One of the advantages of moving mesh methods for the numerical solution of partial differential equations is their ability to track moving boundaries. In this thesis we propose a velocity-based moving mesh method in which we primarily focus on moving the nodes so as to preserve local mass fractions. To recover the solutions from the mesh we use an integral approach which avoids altering the structure of the original equations when incorporating the velocity. We apply our method to a range of moving boundary problems: the porous medium equation; Richards' equation; the Crank-Gupta problem; an avascular tumour growth model. We compare the numerical results to exact solutions where possible, or to results obtained from other methods, and find that our approach is accurate. We apply three different strategies to the tumour growth model, using information from the previous chapters, which enables us to make comparisons between the different approaches. We conclude that our moving mesh method can offer equal accuracy and better resolution, whilst offering greater flexibility than a standard fixed mesh approach.

Declaration

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

Tamsin Lee

Acknowledgments

I would like to thank EPRSC for funding my studies. I extend that gratitude to Reading University Maths Department who organised everything from the funding to endless monitoring meetings, especially Sue Davis, the heart of the department. I thank Peter Grindrod and Paul Glaister for making my monitoring meetings a comfortable environment to discuss my work and help see the direction of my thesis. An additional thanks to Peter Grindrod for his contributions to my thesis. I am also grateful to Marcus Tindall for the aid of his enthusiastic insight into tumour modelling that has been invaluable. In addition, I thank Matthew Hubbard for suggesting the tumour growth problem.

The most important gratitude is for my supervisors, Mike Baines and Steve Langdon, for their continuing patience, and ensuring that studying a Ph.D. with them was an enjoyable experience. However, a Ph.D. is not all about fun, so I am additionally grateful that Mike and Steve have resulted in me continually achieving work I am proud of. I am particularly grateful to Mike, but trying to express the enormity of my gratitude would be like trying to divide by zero.

A big thanks to Bonhi for being a ‘writing-up-veteran’ that I could turn to frequently for advice, and for injecting some enthusiasm and pride into my work when it was waning. I also thank Sanita for the delight of sharing an office with her for so many years. The Reading University Maths Department is crammed with exceptionally lovely staff and students, and I feel fortunate to have been part of it. However, it is such a nice environment to have a chat and a giggle that I’ve had to work from home more. This would not have been possible without my mum kindly buying me a laptop. Working from home, though, has led me to turn Mark’s living environment into the crime scene of a small forest. Not only has he had to endure maths notes in every corner of his flat, but he’s patiently listened to me give him regular updates about every twist and turn and computer bug. I am thankful for his apparent interest in a topic he knows nothing about, and for his generous financial support which has ensured that my hairband collection hasn’t had to suffer too much whist studying. Finally, as with every achievement in my life, I must acknowledge the love and support from my friends and family, in particular, the unwavering and comforting support from Lauren.

Table of Variables

Symbol	Meaning
t	Time variable
x	Independent space variable
$\mathcal{L}$	Spatial differential operator that conserves mass
$\mathcal{G}$	Spatial differential operator that does not conserve mass
$\mathcal{H}$	Spatial differential operator that conserves mass
$S(x,t)$	Source term
$u(x,t)$	The solution to a patial differential operator
$a(t)$,$b(t)$	The boundary positions of the solution
x_j, $j = 0, 1, \ldots, N$	The mesh with N nodes
Δt	Time step
m	Time level such that $t = m\Delta t$
$\tilde{x}_j(t)$	A time-dependent variable that coincides instantaneously with x_j
$\frac{\mathrm{d}\tilde{x}_j}{\mathrm{d}t} = v(\tilde{x}_j(t), t) = \tilde{v}_j(t)$	The velocity of the nodes
$u(\tilde{x}_j(t), t) = \tilde{u}_j(t)$	The solution at the nodes
c_j	The mass of the initial solution in the region $x \in [a(t), \tilde{x}_j(t)]$
c_{j_-}	The mass of the initial solution in the region $x \in [\tilde{x}_{j-1}(t), \tilde{x}_j(t)]$
c_{j_+}	The mass of the initial solution in the region $x \in [\tilde{x}_j(t), \tilde{x}_{j+1}(t)]$
$\theta(t)$	The total mass of the solutionat time t
$\Theta_j(t)$	The mass of the solution at time t between $\tilde{x}_{j-1}(t)$ and $\tilde{x}_{j+1}(t)$
x_j^m	An approximation to $\tilde{x}_j(t^m)$
v_j^m	An approximation to $\tilde{v}_j(t^m)$
u_j^m	An approximation to $\tilde{u}_j(t^m)$
θ^m	An approximation to $\theta(t^m)$
Θ_j^m	An approximation to $\Theta_j(t^m)$
$(\cdot)^m_{j+\frac{1}{2}}$	$\frac{1}{2}[(\cdot)_{j+1} + (\cdot)_j]$
$(\cdot)^m_{j-\frac{1}{2}}$	$\frac{1}{2}[(\cdot)_j + (\cdot)_{j-1}]$
$\Delta(\cdot)^m_j$	$(\cdot)^m_{j+\frac{1}{2}} - (\cdot)^m_{j-\frac{1}{2}}$
$\Delta(\cdot)_{j_-}$	$(\cdot)^m_j - (\cdot)^m_{j-1}$
$\Delta(\cdot)_{j_+}$	$(\cdot)^m_{j+1} - (\cdot)^m_j$

Contents

List of Figures

List of Tables

1

Introduction

Partial differential equations (PDEs) are frequently used to describe physical laws governed by the conservation of mass, energy or momentum. The analytic solution to these PDEs is often difficult to derive, and furthermore, the solution may exhibit behaviour that is challenging to capture numerically. Adaptive numerical schemes change the mesh during the course of computation in response to changes in the dependent variable (or its approximation) to achieve greater accuracy and/or greater efficiency, and possibly also to improve stability. Generally, for the adaptive numerical solution of PDEs either a static or a moving mesh method can be used. For each approach, a discrete solution is initially defined on a given mesh. When using a static mesh, at each time step a new mesh (which may have a different number of nodes to the previous mesh) is generated and the solution is interpolated from the old mesh to the new. Static methods are generally robust, but the computation can be slow.

Adaptive mesh techniques play an important role in improving methods for the numerical solution of PDEs, by concentrating mesh points in areas of interest. An adaptive mesh scheme becomes preferable to a fixed mesh scheme when these areas of interest represent only a fraction of the domain being investigated. Selectively increasing the resolution in these regions is computationally less expensive than refinement of the mesh over the entire grid. Over the past three decades, adaptive algorithms have been applied to phase change problems [8, 14, 19, 70], blow-up problems [27], hyperbolic conservation laws [53], and general classes of time-dependent problems [4, 48, 72].

There are three main types of grid adaptation, the most common type being h-refinement, which uses a static mesh and adds or removes nodes to or from the existing mesh, resulting in local refinement or coarsening of the mesh. Another is p-refinement, where a finite element discretisation of the PDE is used with local polynomials, in which the order of the polynomials is increased or decreased to adapt the method according to the smoothness of the solutions, often as measured by error estimates. It is common to combine these two methods to give hp-methods. However, hp-methods can be complex, need not take advantage of any dynamic properties of the underlying solution, and the error estimates rely heavily on certain assumptions on the solution that may be difficult to verify for strongly nonlinear problems [28]. Our work focuses on the least common type of grid adaptation: r-refinement, or moving mesh methods, which relocate a constant number of mesh points depending on a chosen mechanism. The mechanism can be chosen to manipulate the nodes to give higher resolution in regions of interest, or to correspond to global properties. Such moving meshes are attracting increasing interest, especially in the numerical approximation of time-dependent problems, since the continuous movement of the mesh allows easier inclusion of time integrators. It is apparent from the significant body of research already accumulated that moving grids have much to offer in terms of improved efficiency. The recent book by Huang and Russell [57] presents the theoretical and practical aspects of r-adaptivity, with particular emphasis on its application to time-dependent PDEs.

In this thesis an adaptive finite volume method is presented for the solution of nonlinear time-dependent PDEs with moving boundaries, using a moving mesh. We apply this method only to one-dimensional problems, and consequently refer to it as a finite difference method throughout this thesis. The approach is prompted by recent interest in geometric integration and scale invariance (see for example [23, 24]) which has rekindled interest in the use of adaptive moving meshes. Scale invariance treats independent and dependent variables alike, suggesting that both solution and mesh should be varied simultaneously when designing numerical schemes so as to inherit this property.

When using a moving mesh, a mesh with a fixed number of nodes is sought which moves smoothly with the solution itself. Often, a mesh equation and the differential equation are solved simultaneously to generate the new nodes and solution. With a moving mesh, interpolation of dependent variables from the old mesh to the new mesh is unnecessary.

There are many different approaches to moving the mesh, i.e. to define a mesh equation. The approach taken in this thesis is a one-dimensional finite difference (finite volume) version of the moving mesh finite element approach proposed by Baines, Hubbard and Jimack [5]. As such, the moving mesh equations for mass conserving problems are based on conservation, specifically upon conserving the local proportion of the total integral (mass) of the dependent variable across the domain. We concentrate on one-dimensional finite differences since they offer greater scope for understanding the mechanism involved, and for analysis which may be extrapolated to more general situations. Although not considered here, the integral may be generalised to conserve other quantities [16, 17], yielding an approach similar to that of using a monitor function to control the movement of the mesh; for example, as in the Moving Mesh Partial Differential Equation (MMPDE) method [14, 55]. It is also strongly related to the Deformation method of Liao and co-workers [67, 68] and to the Geometric Conservation Law (GCL) method of Cao, Huang and Russell [33, 94]. These alternative moving mesh methods are discussed in more detail in Chapter 2, with particular focus given to the work of Baines, Hubbard and Jimack [5], since the method in this thesis is more closely related to [5].

In Chapter 3 we develop the finite difference moving mesh method. This method is then applied to a variety of problems where the solution has a moving boundary.

We begin with the porous medium equation (PME) in Chapter 4, since it is the simplest nonlinear diffusion problem which appears in a physically natural way, describing processes involving fluid flow, heat transfer or diffusion. It can also be applied to mathematical biology and other fields. All of these reasons support the interest in its study both for the mathematician and the scientist [99]. There has been extensive research into properties of the PME [11, 99]. We recall how a self-similar solution is derived, and use this as a comparison for our numerical results, making it a suitable choice to demonstrate our moving mesh method. The self-similar solution is geometrically symmetric so we model half the problem which contains only one moving boundary.

In Chapter 5 we generalise the PME to a second problem, Richards' equation, which is used to describe the movement of a fluid through unsaturated soil [61]. Richards' equation is similar to the PME in that it conserves mass; however, it is not symmetric so we model two moving boundaries. We demonstrate that deriving a self-similar solution is more complicated than with the PME. By demonstrating the difficulty of deriving an analytical solution we provide motivation for a numerical approach. Both the PME and Richards' equation conserve mass, making the next natural progression to be a problem that does not conserve mass.

The Crank-Gupta problem is the next moving boundary problem, considered in Chapter 6. This is the first example of a problem where the solution does not conserve mass, since it contains a source term. It was derived to model the diffusion of oxygen through an absorbing tissue [38], but also applies in the Black-Scholes framework of financial modelling, since the valuation of an American option is a free boundary problem similar to the oxygen consumption problem. In the original problem, boundary conditions are set such that there is only one moving boundary. We derive a self-similar series solution, but find that it does not satisfy the set boundary conditions. As with Richards' equation, this demonstrates the need for a numerical solution.

The Crank-Gupta problem has a negative source term, resulting in a solution that decreases in mass. The final problem is covered in Chapter 7 where we consider a model for avascular tumour growth which has an increasing mass. This is the most complicated of the four problems. We begin this chapter with a brief introduction to tumours, and the role that mathematics has played on tumour growth research. We note that the modelling of tumour growth is an area of much interest to mathematical biology. We also find that such models are often numerically solved in one-dimension using a fixed mesh approach. We demonstrate that our moving mesh method can be applied to this problem and compare it to the standard approach. In fact, we compare three moving mesh strategies for this problem and show that our moving mesh method offers a numerical scheme that is accurate and can track features of the tumour.

We end the thesis with a summary and some conclusions from the application of our moving mesh method to these various problems. The original aspects of this work are as follows:

- The Conservation Method using finite differences (finite volumes);
- Using a semi-implicit time-stepping scheme for our moving mesh method;
- Deriving a self-similar series solution for the Crank-Gupta problem;
- Comparisons of different numerical approaches to solve the avascular tumour growth model;
- The Conservation Method with finite elements using a Delaunay triangulation to re-mesh at each time-level.

We begin in Chapter 2 by considering the background of moving mesh methods.

2 Background on Moving Mesh Methods

Moving mesh methods belong to the class of adaptive mesh methods. They are often referred to as r-adaptivity (relocation) methods.

Such methods have a natural application to problems with close coupling between spatial and temporal length scales, such as problems with symmetry, scaling invariance and self-similar solutions [11, 25], where the mesh points become the natural coordinates for an appropriately rescaled problem. However, moving meshes have yet to become part of established numerical codes [28]. In particular, there remain unanswered questions with regards to convergence, the nature of the meshes generated, and the error estimates that can be obtained when using them to solve PDEs with rapidly evolving structures. As recently noted by Budd, Huang, and Russell [28], much of the analysis of such methods has been for one-dimensional problems, including the one-dimensional solvers which make use of r-adaptive methods such as MOVCOL [56, 85] and the continuation code AUTO [41].

Moving mesh methods start with an initial mesh that may or may not be uniform, and then move the nodes whilst the number of nodes (and usually the mesh topology in two-dimensions), remain constant. The target criterion that is used to motivate the movement of the nodes is such that the nodes naturally move to where the solution has ‘interesting behaviour’, such as at a moving boundary or interface, or where ‘blow up’ occurs. The area of interest can often be identified by a rapid variation of either the solution, or one of its derivatives. The error from an r-adaptive method will depend not only on the solution itself, but also the number and position of the mesh nodes. Moving mesh methods often utilise two domains: a computational space and the physical domain in which the underlying equation is posed. The location, or the velocity, of the mesh points is often determined (although not in this thesis) by solving a system of auxiliary PDEs, usually called the moving mesh equations. The feature which is used to determine the mesh movement typically manifests itself as a 'monitor function', which for example, is determined uniquely for each problem to encourage the solution error over each mesh cell to be evenly distributed. The monitor function is usually constructed in one of three ways:

(1) depending upon *a priori solution estimates*, such as arc length or mass;

(2) depending upon *a posteriori error estimates* or the solution residual, as used in moving Finite Element Methods (FEM) [4], or estimates of the derivative jump across element boundaries [92]. An important motivation for the monitor function is the equidistribution principle, first introduced by de Boor for solving boundary value problems for ordinary differential equations (ODEs). The equidistribution principle is a numerical principle which involves selecting mesh points such that some measure of the solution error is equalised over each subinterval;

(3) depending upon on some underlying physics related to the solution, such as the potential temperature or the vorticity in a meteorological problem [24]. In the case of scale-invariant problems such physical estimates are often optimal since the mesh points become the natural coordinates for an appropriately rescaled problem [28].

When using moving mesh methods, extra care is required to preventing mesh tangling and to ensure mesh regularity and isotropy (where relevant).

It is also desirable that discretisations of the underlying PDEs on such meshes (in either the computational or the physical domain) should retain important properties of the underlying physical solution, such as conservation laws and scaling structures [92]. Provided that these conditions are satisfied, r-adaptive methods can be used with considerable success for many time-evolving systems.

Constructions of moving mesh methods vary considerably, and in their final forms the moving mesh equations are very different. In the survey paper [34] moving mesh methods are classed as either velocity-based or location-based. We examine these two classes separately in §2.1 and §2.2, discussing key examples in each class. This thesis uses a velocity-based approach so we shall consider the development of these methods in more detail. In §2.3 we pay particular attention to the finite element Conservation Method of

Baines, Hubbard and Jimack [5, 9], since the methods we will use are essentially finite difference variations on their work. However, we begin with location-based methods.

2.1 Location-based methods

Location-based moving mesh methods are based upon time-dependent mappings which directly control the location of mesh points, or in the continuous sense the time-dependent mapping of the computational coordinate ξ to the physical coordinate x.

As in [34], consider the one-dimensional case of an adaptive mapping $x(\xi, t)$ from a computational domain Ω_c to a physical domain Ω. If the mesh on Ω_c is uniform then $\frac{\partial \xi}{\partial x}$ measures the density of the mesh on Ω. To control the mesh, the mesh density at any given time t is taken to be proportional to a prescribed function $m(x) > 0$, i.e.

$$\frac{\partial \xi}{\partial x} = c\, m(x), \tag{2.1}$$

where c is a constant. This is equivalent to the equidistribution principle for the monitor function $m(x)$. Dividing (2.1) by $m(x)$ and differentiating they obtain

$$\frac{\partial}{\partial x}\left(\left[m(x)\right]^{-1}\frac{\partial \xi}{\partial x}\right) = 0 \text{ on } \Omega,$$

which, given m, can be solved for ξ in terms of x. It is also the Euler-Lagrange equation of the quadratic functional

$$I[\xi] = \int_{\Omega} \left[m(x)\right]^{-1}\left(\frac{\partial \xi}{\partial x}\right)^2 \,\mathrm{d}x,$$

which is useful for generalising to two dimensions [28].

The survey paper [34] gives a brief description of various ways to extend the equidistribution principle (2.1) to higher dimensions. These include the work of Knupp [58, 59], Winslow [105], Thompson et al. [95], Brackbill and Saltzman [20], Dvinsky [45] and Brackbill [21].

A typical location-based moving mesh method is a variational approach which defines the mapping as the minimiser of a functional. According to [55], an early example of this was given by Dorfi and Drury [42], where a separate equation for mesh speeds is developed via a function chosen such that mesh resolution is controlled. A simple relation between the speeds of the mesh points and the function is solved in conjunction with the underlying PDE. Despite no formal mention of equidistribution ideas, Huang et al. [55] note that the Dorfi and Drury method derives the moving mesh equation directly from an equidistribution principle.

Huang et al. [55] recommend that a moving mesh method should not only move nodes to areas of interest, but require a simple algorithm that is easy to program, and be reasonably insensitive to the choice of user-defined parameters. In their work this is achieved by constructing moving mesh equations directly from the numerical equidistribution principle. Furthermore, they recommend that in order to allow ease of comparison and theoretical analysis of this moving mesh method, the moving mesh equations should have a continuous form, and in [55] these equations are devised and referred to as moving mesh partial differential equations (MMPDEs). MMPDE-based methods are a major class of location-based moving mesh methods, especially for one-dimensional problems [55, 70], and have been incorporated into codes such as those mentioned in the Introduction, MOVCOL and AUTO [28].

Further location-based moving mesh methods include optimal transport methods [26], which are a natural generalisation of MMPDE methods in one dimension [28]. The key idea behind an optimal mesh is that it should be one which is closest to a uniform mesh in a suitable norm, consistent with satisfying the equidistribution principle. Budd and co-workers demonstrate that this can lead to a Monge-Ampere equation defining the mapping from the computational to the physical domain [26, 28].

We now turn to velocity-based moving mesh methods, which is the approach used in this thesis. In this approach the boundary velocity is generated automatically, making a velocity-based approach particularly suited to the moving boundary problems we consider in this thesis. In addition, a velocity-based approach does not require reference to a computational space and, compared to a location-based approach, the velocity is easier to insert into the PDE in a moving frame.

2.2 Velocity-based methods

Velocity-based methods use a Lagrangian (moving) co-ordinate system to directly provide a mesh velocity. The velocity of the nodes may be defined in a variety of ways, and many constructions have appeared in the literature, some mathematically based and some based on physical analogies. The mathematical approaches often target a mapping that equidistributes a monitor function, and may use residual minimisation, error estimates, or geometric considerations, whereas physical approaches usually rely on ideas from Lagrangian fluid dynamics or mechanical analogies. Mesh movement may also be induced entirely by the normal velocities of the boundaries in conjunction with averaging techniques. Solution features are often assumed to be convected with the flow, and it is natural to evolve the mesh points to follow the flow itself, [28]. This intuitive movement of mesh points makes a velocity-based moving mesh approach particularly suited for fluid flow, but the approach is more general.

In this section a number of velocity-based methods are described. We begin in §2.2.1 with schemes related to fluid dynamics which use a purely Lagrangian approach. This leads on to methods which rely on the so-called ALE (Arbitrary Lagrangian-Eulerian) formulation, which often uses velocities generated by mechanical analogies, see §2.2.2.

In §2.2.3 we turn to mathematically motivated constructions. We briefly describe the Moving Finite Element method [72, 73], which was the first method to determine the mesh and the solution simultaneously. Then, in §2.2.4 a derivation of the GCL method is presented, as described in [33], which is based on conservation of a key variable. Finally the Conservation Method [5, 9] is introduced. Since this is the approach we shall use in the rest of this thesis (with finite differences instead of finite elements), we expand on this method in detail in §2.3.

2.2.1 Fluid dynamics

In classical theoretical fluid dynamics the motion of fluids may be described by taking either the Lagrangian or Eulerian point of view. In the Lagrangian description, each moving fluid particle (with its attributes) is followed individually and is identified by its initial position, whereas, in the Eulerian description variables such as density and velocity are evaluated at fixed locations.

A link between the two is provided by the Reynolds Transport Theorem [104] in the form

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega(t)} u\,\mathrm{d}\boldsymbol{x} &= \int_{\Omega(t)} \frac{\partial u}{\partial t}\,\mathrm{d}\boldsymbol{x} + \oint_{\partial\Omega(t)} u\boldsymbol{v}\cdot\hat{\boldsymbol{n}}\,\mathrm{d}S, \qquad (2.2)\\ &= \int_{\Omega(t)} \left\{\frac{\partial u}{\partial t} + \nabla\cdot(u\boldsymbol{v})\right\}\,\mathrm{d}\boldsymbol{x}, \qquad (2.3)\end{aligned}$$

(by the Divergence Theorem) for a general moving region $\Omega(t)$, where u is the fluid density, $\boldsymbol{v}$ is the fluid velocity, and $\hat{\boldsymbol{n}}$ is the unit outward normal. Here $\mathrm{d}S$ is an element of the boundary $\partial\Omega(t)$ of $\Omega(t)$. The theorem states that the rate of change of mass in a moving frame comprises the rate of change in a fixed frame plus the flux of u across the boundary.

In the Lagrangian description, conservation of mass is expressed as

$$\int_{\Omega(t)} u\,\mathrm{d}\boldsymbol{x} = \text{ constant in time,}$$

whilst in the Eulerian description the mass conservation equation is

$$\frac{\partial u}{\partial t} + \nabla \cdot (u\boldsymbol{v}) = 0. \tag{2.4}$$

The equivalence of the two is inherent in (2.3), remembering that this equation holds for all $\Omega(t)$.

In recent times, computational fluid dynamics (CFD) has sought to describe the motion of fluids numerically by both approaches. By far, the most common approach has been through discretisations of the Eulerian description, where the equations of motion are discretised on a fixed mesh. The same can be said for the numerical solutions of PDEs in general. However, Lagrangian moving mesh methods can play a substantial role in obtaining high resolution solutions to problems with implicit moving boundaries or singularities, and in mimicking invariance properties. Use of the Lagrangian description has been less common and largely confined to problems where surfaces and interfaces are of primary importance, e.g. [74, 82, 102]. The natural discretisation of the Lagrangian approach is to follow the velocities of the fluid particles using a moving mesh, but compromises usually have to be made, often with regards to the tendency of the mesh to tangle and lose its character. For example, Harlen et al. [1, 52] apply a Lagrangian approach to the solution of convection equations arising in the simulation of viscoelastic flows. Although the nodes of the finite element mesh are transported with the fluid particles, the mesh itself has to be reconnected after each time-step in order to maintain the Delaunay property [15]. These, and other considerations have prompted the use of the so-called ALE methods (see §2.2.2), where local modifications of the Lagrangian velocities are made as the computation develops [13, 35]. This framework can be used with any imposed velocity and is of particular relevance to velocity-based approaches [7].

2.2.2 ALE (Arbitrary Lagrangian Eulerian) methods

Consider the generic PDE

$$\frac{\partial u}{\partial t} = \mathcal{L}u, \tag{2.5}$$

where $\mathcal{L}$ is a spatial partial differential operator. The Reynolds Transport Theorem in the form (2.2) allows the solution u of (2.5) to be obtained in a frame moving with any given velocity via the ALE equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega(t)} u \,\mathrm{d}\boldsymbol{x} = \int_{\Omega(t)} \mathcal{L}u \,\mathrm{d}\boldsymbol{x} + \oint_{\partial\Omega(t)} u\boldsymbol{v}\cdot\hat{\boldsymbol{n}}\,\mathrm{d}S.$$

This equation is widely used in the numerical solution of fluid-structure interaction problems, for example in [54, 60, 71, 86]. The specific mechanism for constructing the mesh velocities varies significantly, from treating the mesh as though it is a physical material with its own constitutive law [60] through to defining the mesh motion purely with the goal of optimizing geometric qualities of the mesh [86].

Other ALE forms, such as the differential form,

$$\frac{Du}{Dt} = \frac{\partial u}{\partial t} + \boldsymbol{v} \cdot \nabla u, \tag{2.6}$$

have also been used for free-surface problems, based upon maintaining mesh equality [78, 79], Laplacian smoothing [89] or pseudo-solid deformation [32, 101]. Other applications which have benefited from successful ALE algorithms include phase-change problems [70, 93], and the interaction of free surfaces with solid boundaries [3, 91, 100]. However, not all ALE methods need to be applied to, or driven by, fluid flow problems.

2.2.3 Moving finite elements

Where there is no specific physical motivation for assigning a velocity to each node of the finite element mesh (unlike Lagrangian-based methods for equations of fluid flow, for example), some other mechanism for determining an appropriate mesh velocity field is required. A well-known approach is the moving finite element (MFE) method of Miller and Miller [72] and Miller [73]. For the time-dependent PDE (2.5), the continuous version of the MFE method determines the solution and mesh simultaneously by minimising a discrete residual of the PDE, $u_t - \mathcal{L}u$, in a moving frame, i.e. finding $\min_{v, \frac{Du}{Dt}} I$ where

$$I\left[\boldsymbol{v}, \frac{Du}{Dt}\right] \equiv \int_\Omega \left(\frac{Du}{Dt} - \nabla u \cdot \boldsymbol{v} - \mathcal{L}u\right)^2 W \,\mathrm{d}\boldsymbol{x}, \tag{2.7}$$

and $\frac{Du}{Dt}$ is defined by (2.6). We observe that the term in the brackets on the right-hand side is the square of a weighted residual of (2.6). The weight function, $W = 1$ was used in the original version of MFE as described in [72, 73]. Later, a gradient weighted version of MFE (GWMFE) [36, 37] was introduced, which uses

$$W \equiv \frac{1}{1 + \|\nabla u\|^2}.$$

Observe that the MFE method advects the mesh at the same time as the solution. In practice, an extended Galerkin method is used, arising from a discretised form of (2.7).

This method is elegant, and when tuned correctly, works well [4, 36, 37]. However, in practice, in order to minimise I one has to construct normal equations which can become singular and significant regularisation is needed.

2.2.4 The Geometric Conservation Law (GCL)

The GCL is a tool which has been used for many years in the engineering community to develop cell-volume-preserving finite-volume schemes. An early example, formulated in [97], is the Space Conservation Law (SCL) which was approximated in the simulation of fluid flow on moving meshes along with conservation of mass, momentum and energy. It was later resurrected by Thomas and Lombard [94], who termed it the Geometric Conservation Law (GCL), as a constraint such that any numerical scheme applied on a moving mesh should reduce to the GCL when the solution is constant, i.e. the movement of the mesh should not create nor destroy 'space'. Writing the physical PDEs in conservative form, Thomas and Lombard update the Jacobian at a new time based on the GCL, and accurate results are obtained for implicit finite-difference and finite-volume solutions of unsteady and steady supersonic flow equations. The desire to satisfy a GCL favours the finite volume framework, which provides the most natural formulation of the GCL due to its inherent integral conservation properties. A similar approach is used in this thesis. Classic examples are [43, 44] where the GCL is incorporated into a finite-volume procedure, and is applied to a number of fluid flow test cases, including the Navier-Stokes equations.

In studies by Cao, Huang and Russell [33] and Baines, Hubbard and Jimack [5, 9] we see the GCL being used as an algorithmic device to obtain mesh velocities, rather than guaranteeing mesh quality. In particular, the GCL can be used to eliminate the use of the Jacobian that relates the computational domain Ω_c to the physical domain Ω so as to relate the mesh velocities directly to a monitor function.

Following [94], to obtain the integral form of the GCL let A_c be an arbitrary, fixed cell in the computational domain Ω_c enclosed by a smooth boundary ∂A_c, and let $A(t) = \{\boldsymbol{x} | \boldsymbol{x} = \tilde{\boldsymbol{x}}(\boldsymbol{\xi}, t) \, \forall \boldsymbol{\xi} \in A_c\}$ be the corresponding cell in the physical domain Ω under the time-dependent coordinate transformation $\boldsymbol{x} = \tilde{\boldsymbol{x}}(\boldsymbol{\xi}, t)$. Then the change in volume of $A(t)$ equals the total flux through the surface $\partial A(t)$, i.e.

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{A(t)} \mathrm{d}\boldsymbol{x} = \int_{\partial A(t)} \boldsymbol{v} \cdot \, \mathrm{d}\boldsymbol{S}, \tag{2.8}$$

where $\boldsymbol{v}$ is the mesh velocity. This is the integral form of the GCL [94], which is simply the Reynolds Transport Theorem (2.2) with constant integrand. Using the change of variables defined by the coordinate transformation $\boldsymbol{x} = \tilde{\boldsymbol{x}}(\boldsymbol{\xi}, t)$, the left-hand side of (2.8) can be rewritten as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{A(t)} \mathrm{d}\boldsymbol{x} = \frac{\mathrm{d}}{\mathrm{d}t}\int_{A_c} J(\boldsymbol{\xi}, t)\ \mathrm{d}\boldsymbol{\xi} = \int_{A_c} \frac{D}{Dt} J(\boldsymbol{\xi}, t)\ \mathrm{d}\boldsymbol{\xi}, \tag{2.9}$$

where $\frac{D}{Dt}$ is given by (2.6) and denotes the time derivative in the coordinate system $(\boldsymbol{\xi}, t)$. In addition, using the Divergence Theorem, (2.8) can be written as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{A(t)} \mathrm{d}\boldsymbol{x} = \int_{\partial A(t)} \boldsymbol{v}\cdot\ \mathrm{d}\boldsymbol{S} = \int_{A(t)} \nabla\cdot\boldsymbol{v}\ \mathrm{d}\boldsymbol{x} = \int_{A_c} (\nabla\cdot\boldsymbol{v}) J(\boldsymbol{\xi}, t)\ \mathrm{d}\boldsymbol{\xi}. \tag{2.10}$$

Noting that A_c is arbitrary, from (2.9) and (2.10), a differential form of the GCL

$$\nabla\cdot\boldsymbol{v} = \frac{1}{J}\frac{DJ}{Dt},$$

is obtained, which is instrumental in deriving GCL moving mesh methods. It is stated in [33] that a simple and straightforward way to use the Jacobian for mesh adaptation is to use the equidistribution equation

$$J(\boldsymbol{\xi}, t) = \frac{c(\boldsymbol{\xi}, t)}{m(\boldsymbol{x}(\boldsymbol{\xi}, t), t)}, \tag{2.11}$$

where $m = m(\boldsymbol{x}, t) > 0$ is a user-defined monitor function, the size of which reflects the local difficulty in approximating the solution of the underlying problem, and $c = c(\boldsymbol{\xi}, t)$ is a time-dependent function determined by the initial coordinate transformation. In one dimension (2.11) is precisely the well-known equidistribution principle, when c is independent of $\boldsymbol{\xi}$ [55], so it can be viewed as a generalisation of this principle. It is worth mentioning that the Deformation method [67, 68, 88], which is a velocity-based method that takes a strongly geometrical approach, is a special case of the GCL [33].

2.2.5 The Conservation Method

The Conservation Method of Baines, Hubbard and Jimack [5, 6, 7] is built on the same foundations as the GCL method of Cao, Huang and Russell [33]. Instead of recasting the GCL as a minimisation problem for the purpose of discretisation, this method uses the conservative (integral) form to find the discrete velocities directly. The method can be summarised as follows: The PDE is used in conjunction with the Eulerian conservation equation (2.4) to generate a velocity, which is then used to move the mesh in a Lagrangian manner. In simple cases the solution of the PDE can be recovered algebraically *a posteriori* from the Lagrangian form of the same conservation law.

2.3 The Finite Element Conservation Method

Consider a general scalar initial boundary value problem (IBVP) for which the solution $u(\boldsymbol{x}, t)$ conserves mass, where $t > t^0$ and $\boldsymbol{x} \in \Omega(t) \subset \mathbb{R}^d$ (d being the number of dimensions) represent the temporal and spatial variables respectively with t^0 representing the initial time. Let the solution $u(\boldsymbol{x}, t)$ be positive and satisfy the PDE

$$\frac{\partial u}{\partial t} = \mathcal{L}u, \quad \text{in } \Omega(t), \tag{2.12}$$

where $\mathcal{L}$ is a differential operator involving only space derivatives. Zero Dirichlet boundary conditions

$$u(\boldsymbol{x}, t) = 0 \qquad \text{on } \partial\Omega(t),$$

are assumed, where $\partial\Omega(t)$ denotes the boundary of the time-dependent region $\Omega(t)$, and initial conditions

$$u(\boldsymbol{x}, t^0) = u^0(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega(0),$$

where $\Omega(0)$ is the initial domain. Subsequently, $\Omega(t)$ moves with a velocity $\boldsymbol{v}$ to be determined.

Since mass is conserved,

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega(t)} u(\boldsymbol{x}, t) \, \mathrm{d}\boldsymbol{x} = 0.$$

In [5] a weighted total mass is introduced such that

$$c_i = \int_{\Omega(t)} w_i(\boldsymbol{x}, t) u(\boldsymbol{x}, t), \, \mathrm{d}\boldsymbol{x} \qquad i = 0, 1, \ldots, N, \tag{2.13}$$

is constant in time, where w_i is a member of a set of weight functions (which form a partition of unity i.e. $\sum_{i=0}^{N} w_i = 1$) moving with the velocity $\boldsymbol{v}$ of the points of the region $\Omega(t)$. The Conservation Method uses (2.13) in conjunction with a weighted form of the PDE to enable a finite element approach. We begin the description by determining this weighted form of the general PDE, and then demonstrate how [5] derive weighted expressions for the mesh velocity and the updated solution. At the end of this subsection we describe how a finite element formulation can be used to give discrete versions of these expressions.

2.3.1 Determining a weighted form of the general PDE

Since the weight functions w_i move with velocity $\boldsymbol{v}$, they satisfy the advection equation

$$\frac{\partial w_i}{\partial t} + \boldsymbol{v} \cdot \nabla w_i = 0, \qquad i = 0, 1, \ldots, N. \tag{2.14}$$

Differentiating the *weighted* mass (2.13) with respect to time, by applying Reynolds Transport Theorem (2.2) to $w_i u$, and using the Divergence Theorem, we have

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega(t)} w_i u \, \mathrm{d}\boldsymbol{x} = \int_{\Omega(t)} \left(\frac{\partial}{\partial t}(w_i u) + \nabla \cdot \big(w_i u \boldsymbol{v}\big) \right) \mathrm{d}\boldsymbol{x}.$$

is obtained for $i = 0, 1, \ldots, N$. Expanding each term on the right-hand side,

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega(t)} w_i u \, \mathrm{d}\boldsymbol{x} &= \int_{\Omega(t)} \left[w_i \frac{\partial u}{\partial t} + u \frac{\partial w_i}{\partial t} + w_i \nabla \cdot (u\boldsymbol{v}) + u\boldsymbol{v} \cdot \nabla w_i \right] \mathrm{d}\boldsymbol{x}, \\ &= \int_{\Omega(t)} \left[w_i \frac{\partial u}{\partial t} + w_i \nabla \cdot (u\boldsymbol{v}) + u \left(\frac{\partial w_i}{\partial t} + \boldsymbol{v} \cdot \nabla w_i \right) \right] \mathrm{d}\boldsymbol{x}, \end{aligned}$$

for $i = 0, 1, \ldots, N$. The last term vanishes, from (2.14), and hence

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega(t)} w_i u \, \mathrm{d}\boldsymbol{x} - \int_{\Omega(t)} w_i \nabla \cdot (u\boldsymbol{v}) \, \mathrm{d}\boldsymbol{x} = \int_{\Omega(t)} w_i \frac{\partial u}{\partial t} \, \mathrm{d}\boldsymbol{x}.$$

Substituting for $\frac{\partial u}{\partial t}$ on the right-hand side, and using a weighted form of the PDE (2.12), a weak form of the PDE, in integral form, in the moving frame is found as

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega(t)} w_i u \, \mathrm{d}\boldsymbol{x} - \int_{\Omega(t)} w_i \nabla \cdot (u\boldsymbol{v}) \, \mathrm{d}\boldsymbol{x} = \int_{\Omega(t)} w_i \mathcal{L} u \, \mathrm{d}\boldsymbol{x}, \tag{2.15}$$

for $i = 0, 1, \ldots, N$. Equation (2.15) is used in conjunction with the distributed conservation principle (2.13) to derive the mesh velocity $\boldsymbol{v}(\boldsymbol{x}, t)$.

2.3.2 Determining the mesh velocity

To derive an expression for the mesh velocity, (2.13) is differentiated with respect to time, giving

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega(t)} w_i u \, \mathrm{d}\boldsymbol{x} = 0, \quad i = 0, 1, \ldots, N. \tag{2.16}$$

Substituting (2.16) into (2.15) gives

$$-\int_{\Omega(t)} w_i \nabla\cdot(u\boldsymbol{v})\,\mathrm{d}\boldsymbol{x} = \int_{\Omega(t)} w_i \mathcal{L}u\,\mathrm{d}\boldsymbol{x}, \qquad i = 0,1,\dots,N. \tag{2.17}$$

To determine $\boldsymbol{v}(\boldsymbol{x},t)$ uniquely from (2.17), an additional condition is required on the mesh velocity. As in [33], if the vorticity is specified, together with suitable boundary conditions, then the velocity is uniquely determined by Helmholtz Theorem (noting that $u > 0$). For simplicity, we suppose here that the vorticity is zero so that $\nabla\times\boldsymbol{v} = 0$; then there exists the velocity potential ψ, such that

$$\boldsymbol{v} = \nabla\psi. \tag{2.18}$$

Substituting this into (2.17) gives

$$-\int_{\Omega(t)} w_i \nabla\cdot(u\nabla\psi)\,\mathrm{d}\boldsymbol{x} = \int_{\Omega(t)} w_i \mathcal{L}u\,\mathrm{d}\boldsymbol{x}.$$

Applying Green's Theorem to the integral on the left-hand side gives

$$-\oint_{\partial\Omega(t)} w_i u\nabla\psi\cdot\hat{\mathbf{n}}\,\mathrm{d}S + \int_{\Omega(t)} u\nabla\psi\cdot\nabla w_i\,\mathrm{d}\boldsymbol{x} = \int_{\Omega(t)} w_i\mathcal{L}u\,\mathrm{d}\boldsymbol{x}, \tag{2.19}$$

where $\mathrm{d}S$ is an element of the boundary $\partial\Omega(t)$ of $\Omega(t)$. The first term vanishes due to the boundary condition. Equation (2.19) can be used to obtain a unique velocity potential ψ (to an added constant), when $u(\boldsymbol{x},t) > 0$ is known.

The velocity $\boldsymbol{v}(\boldsymbol{x},t)$ may be calculated from ψ using a weak form of (2.18),

$$\int_{\Omega(t)} w_i\boldsymbol{v}\,\mathrm{d}\boldsymbol{x} = \int_{\Omega(t)} w_i\nabla\psi\,\mathrm{d}\boldsymbol{x}, \tag{2.20}$$

which may be necessary if $\boldsymbol{v}$ and $\nabla\psi$ belong to different (non-conforming) spaces.

2.3.3 Recovering the solution

Since c_i remains constant, the expression (2.13) at time t equates to (2.13) at the initial time $t = t^0$, giving

$$\int_{\Omega(t)} w_i(\boldsymbol{x},t)u(\boldsymbol{x},t)\,\mathrm{d}\boldsymbol{x} = \int_{\Omega(t^0)} w_i(\boldsymbol{x},t^0)u(\boldsymbol{x},t^0)\,\mathrm{d}\boldsymbol{x}, \tag{2.21}$$

which is used to determine the solution $u(\boldsymbol{x},t)$ at any given time, since the right-hand side is known from the initial conditions.

The finite element formulation and the full algorithm

In [5], a finite element mesh is set up and the initial condition is projected on to the mesh. Piecewise linear basis function are used, replacing w_i with ϕ_i, where ϕ_i are linear finite element basis functions on the mesh of nodes $\boldsymbol{x}_i(t)$, $i = 0, 1, \ldots, N$, within a polygonal approximation $\bar{\Omega}(t)$ of $\Omega(t)$.

The equations which determine $\nabla\psi$, $\boldsymbol{v}(\boldsymbol{x}, t)$ and $\boldsymbol{u}(\boldsymbol{x}, t)$ (from equations (2.19), (2.20) and (2.21) respectively), are used with $\boldsymbol{x}$, ϕ, $\boldsymbol{v}$, θ and $\boldsymbol{u}$ replaced with piecewise linear approximations. The algorithm is as follows:

Given the mesh and solution at a given time:

- Compute the mesh velocity from the finite element form of (2.19);
- Using a time-stepping scheme, compute the updated mesh;
- Compute the updated solution from the finite element form of (2.21).

In [5]–[9] the algorithm is extended to problems which do not conserve mass, and the use of monitor functions. Results are presented for a number of moving boundary problems, including phase change problems [8]

This thesis applies the above principles used by Baines, Hubbard and Jimack, but whereas they use a finite element method, we investigate how these principles can be applied with finite differences. Although using finite differences essentially limits the scope of the PDEs to be solved to one dimension, nonetheless there are many one-dimensional PDEs which can be accurately solved using a finite difference moving mesh. Furthermore, with finite differences in one dimension, there is scope for making advances in the method and novel analysis which may suggest new routes in two dimensions.

3

A Finite Difference Velocity-Based Moving Mesh Method Based on Conservation

In this chapter we describe a moving mesh approach for one-dimensional IBVPs with moving boundaries, where we use finite differences throughout. In §3.1 we consider problems that conserve mass and derive a mesh that preserves partial masses. In §3.2 we generalise this method by preserving relative partial masses so that it can be applied to problems which do not conserve mass. In §3.3, we consider a moving mesh approach which is relevant to IBVPs that contain a source term since partial masses are balanced with the source term. For each moving mesh approach we derive an expression for the mesh velocity (which is used with a time-stepping scheme to determine an updated mesh), and a formula for the solution on the updated mesh (which is purely algebraic in the first two cases). In §3.4 we investigate the explicit and semi-implicit time-stepping schemes that are applied in this thesis. To ensure that mesh tangling does not occur, we take small time steps when using an explicit time-stepping scheme. When using a semi-implicit scheme, which allows larger time-steps, we prove that the mesh remains monotonic.

3.1 Mass conserving problems

We first describe a velocity-based moving mesh method to numerically solve an IBVP with moving boundaries which conserves mass, with $t > t^0$ and $x \in [a(t), b(t)]$ representing the temporal and spatial variables respectively, and t^0 represents the initial time. The solution $u(x,t)$ of the IBVP satisfies the generic PDE

$$\frac{\partial u}{\partial t} = \mathcal{L}u, \tag{3.1}$$

where $\mathcal{L}$ is a purely spatial differential operator whose exact form will be defined in each application, with zero Dirichlet boundary conditions,

$$u = 0 \quad \text{at} \quad x = a(t), b(t), \quad t > t^0. \tag{3.2}$$

The problem has initial conditions

$$u(x, t^0) = u^0(x). \tag{3.3}$$

As mass is conserved, we can calculate the constant total mass

$$c = \int_{a(t)}^{b(t)} u(x,t)\,\mathrm{d}x, \tag{3.4}$$

from the initial data.

Remark 3.1.1 *We consider only IBVPs for which $u(x,t) > 0$, $x \in (a(t), b(t))$ holds for all $t \geq 0$. This is important for the definition of the algorithm.*

For the problem (3.1)–(3.3), x is an independent variable. We consider a mesh given by x_j, $j = 0, \ldots, N$, such that

$$a(t) \approx x_0(t) < x_1(t) < \ldots < x_{N-1}(t) < x_N(t) \approx b(t). \tag{3.5}$$

We later introduce the dependent variable $\tilde{x}_j(t)$ to represent the $N+1$ nodes of the mesh, which coincide instantaneously with x_j and vary with t. We define the velocity of the j-th node to be

$$v(\tilde{x}_j(t), t) = \tilde{v}_j(t) = \frac{\mathrm{d}\tilde{x}_j}{\mathrm{d}t}.$$

Our moving mesh method moves the nodes such that the partial masses of the solution are conserved, i.e. we determine $\tilde{x}_j(t)$ $(j = 0, \dots, N)$ from

$$c_j = \int_{a(t)}^{\tilde{x}_j(t)} u(x,t)\,\mathrm{d}x, \tag{3.6}$$

where the c_j $(j = 0, \dots, N)$ remain constant in time. Equation (3.6) is consistent with equation (3.4). The c_j are positive since $u > 0$, and $c_N = c$.

For a given mesh $\tilde{x}_j(t)$ and solution $\tilde{u}_j(t) = u(\tilde{x}_j(t), t)$, we now compute the mesh velocity $\tilde{v}_j(t) = v(\tilde{x}_j(t), t)$, and subsequently the updated mesh, to ultimately recover the updated solution on the new mesh. Details are given in following subsections.

3.1.1 Determining the mesh velocity

We obtain an expression for the mesh velocity by differentiating (3.6) with respect to time, giving

$$0 = \frac{\mathrm{d}}{\mathrm{d}t} \int_{a(t)}^{\tilde{x}_j(t)} u(x,t)\,\mathrm{d}x. \tag{3.7}$$

Applying the Leibnitz integral rule to the right-hand side of (3.7) gives

$$0 = \int_{a(t)}^{\tilde{x}_j(t)} \frac{\partial u}{\partial t}\,\mathrm{d}x + \tilde{u}_j(t)\tilde{v}_j(t) - \tilde{u}_0(t)\tilde{v}_0(t).$$

Substituting $\frac{\partial u}{\partial t}$ from the PDE (3.1), and using the boundary conditions (3.2),

$$\int_{a(t)}^{\tilde{x}_j(t)} \mathcal{L}u\,\mathrm{d}x + \tilde{u}_j(t)\tilde{v}_j(t) = 0.$$

Thus, recalling Remark 3.1.1, since $\tilde{u}_j(t) > 0$ in the interior, the interior nodes move such that

$$\tilde{v}_j(t) = -\frac{1}{\tilde{u}_j(t)} \int_{a(t)}^{\tilde{x}_j(t)} \mathcal{L}u\,\mathrm{d}x, \tag{3.8}$$

which holds for $j = 1, \dots, N-1$. The mesh velocities at the boundaries, $\tilde{v}_0(t), \tilde{v}_N(t)$, are extrapolated from the internal mesh velocities.

For any given $\mathcal{L}$, the integral in (3.8) may be approximated using quadrature, to give a discrete form of (3.8) such that $\tilde{v}_j(t)$ is given in terms of $\tilde{x}_j(t)$ and $\tilde{u}_j(t)$.

3.1.2 Advancing the mesh in time

We choose a time step $\Delta t > 0$ and define time-levels $t^m = m\Delta t$, $m = 0, 1, \ldots$, denoting $\tilde{x}_j(t^m)$ by x_j^m. We also use the approximations $u_j^m \approx \tilde{u}_j(t^m)$ and $v_j^m \approx \tilde{v}_j(t^m)$. The updated x_j^{m+1} are calculated using the mesh velocity v_j^m with a time-stepping scheme.

3.1.3 Recovering the solution

To recover the solution on the new mesh we use an incremental form of (3.6),

$$c_{j+1} - c_{j-1} = \int_{x_{j-1}^{m+1}}^{x_{j+1}^{m+1}} u(x,t)\,\mathrm{d}x = \int_{x_{j-1}^{t^0}}^{x_{j+1}^{t^0}} u(x,t^0)\,\mathrm{d}x, \qquad j = 1, \ldots, N-1, \tag{3.9}$$

(where each c_j is a constant known from initial conditions). We approximate the integrals of (3.9) using a simple quadrature rule, which allows us to recover u_j^{m+1}, $j = 1, \ldots, N-1$ on the new mesh. We examine two different quadrature rules, a mid-point approximation and an interpolating approximation on a non-uniform mesh.

A simple mid-point approximation of (3.9) gives

$$(x_{j+1}^{m+1} - x_{j-1}^{m+1})u_j^{m+1} = (x_{j+1}^0 - x_{j-1}^0)u_j^0,$$

which means that the updated solution can be recovered on the new mesh as

$$u_j^{m+1} = \frac{x_{j+1}^0 - x_{j-1}^0}{x_{j+1}^{m+1} - x_{j-1}^{m+1}} u_j^0, \tag{3.10}$$

for $j = 1, \cdots, N-1$. The solution at the boundaries, u_0^{m+1}, u_N^{m+1}, is determined by the boundary conditions (3.2). Note that at any given time-level the discrete partial mass $(x_{j+1}^{m+1} - x_{j-1}^{m+1})u_j^{m+1}$ is preserved, and is equal to the corresponding initial partial mass $(x_{j+1}^0 - x_{j-1}^0)u_j^0$. This approximation is second order accurate on a uniform mesh, but only first order on a non-uniform mesh.

As the mesh is not necessarily uniform, it is more accurate to generalise equation (3.10) so that it is fully second order, i.e. exact for a linear solution on a *non-uniform* mesh. Now, if $u(x,t)$ were linear in the interval $(\tilde{x}_{j-1}, \tilde{x}_{j+1})$, then it would hold that

$$u(x,t) = u(\tilde{x}_j, t) + (x - \tilde{x}_j)g(\tilde{x}_j, t), \qquad x \in (\tilde{x}_{j-1}, \tilde{x}_{j+1}), \tag{3.11}$$

where $g(\tilde{x}_j, t)$ is the slope of of u at x_j.

We define the incremental differences between the constants (in time) c_j, $j = 1, \ldots, N-$

1 in the intervals adjacent to x_j as

$$\begin{aligned} c_{j_-} &= c_j - c_{j-1}, \\ c_{j_+} &= c_{j+1} - c_j, \end{aligned}$$

where each of these are determined from the initial conditions. We note that $c_{j_+} + c_{j_-} = c_j$, from (3.9). We then use (3.11) to derive two equations for the sum $(c_{j_+} + c_{j_-})$ and difference $(c_{j_+} - c_{j_-})$. These equations are evaluated at $t = t^{m+1}$, and the unknown slope $g_j^{m+1} \approx g(\tilde{x}_j, t^{m+1})$ is eliminated, so that we achieve an expression for u_j^{m+1} which is exact for linear u. The sum is given by

$$\begin{aligned} c_{j_+} + c_{j_-} &= \int_{\tilde{x}_{j-1}(t)}^{\tilde{x}_{j+1}(t)} u(x,t)\ \mathrm{d}x, \\ &= \int_{\tilde{x}_{j-1}(t)}^{\tilde{x}_{j+1}(t)} \{u(\tilde{x}_j,t) + (x - \tilde{x}_j) g(\tilde{x}_j,t)\}\ \mathrm{d}x, \end{aligned} \tag{3.12}$$

and the difference by

$$c_{j_+} - c_{j_-} = \int_{\tilde{x}_j(t)}^{\tilde{x}_{j+1}(t)} \{u(\tilde{x}_j,t) + (x - \tilde{x}_j) g(\tilde{x}_j,t)\}\ \mathrm{d}x - \int_{\tilde{x}_{j-1}(t)}^{\tilde{x}_j(t)} \{u(\tilde{x}_j,t) + (x - \tilde{x}_j) g(\tilde{x}_j,t)\}\ \mathrm{d}x. \tag{3.13}$$

Evaluating the integrals of (3.12) and (3.13) at $t = t^{m+1}$ gives

$$\begin{aligned} c_{j_+} + c_{j_-} &= u_j \left[x_{j+1}^{m+1} - x_{j-1}^{m+1}\right] + \frac{g_j}{2}\left((x_{j+1}^{m+1})^2 - (x_{j-1}^{m+1})^2\right) - g_j x_j (x_{j+1}^{m+1} - x_{j-1}^{m+1}), \\ &= u_j (x_{j+1}^{m+1} - x_{j-1}^{m+1}) + \frac{g_j}{2}\left[\left(\Delta x_{j_+}^{m+1}\right)^2 - \left(\Delta x_{j_-}^{m+1}\right)^2\right], \end{aligned} \tag{3.14}$$

and

$$\begin{aligned} c_{j_+} - c_{j_-} &= u_j^{m+1} \Delta x_{j_+}^{m+1} + \frac{g_j^{m+1}}{2}\left[(x_{j+1}^{m+1})^2 - (x_j^{m+1})^2\right] - g_j^{m+1} x_j^{m+1} \Delta x_{j_+}^{m+1} \\ &- u_j^{m+1} \Delta x_{j_-}^{m+1} - \frac{g_j^{m+1}}{2}\left[(x_j^{m+1})^2 - (x_{j-1}^{m+1})^2\right] + g_j^{m+1} x_j^{m+1} \Delta x_{j_-}^{m+1}, \\ &= u_j^{m+1}\left[\Delta x_{j_+}^{m+1} - \Delta x_{j_-}^{m+1}\right] + \frac{g_j^{m+1}}{2}\left[\left(\Delta x_{j_+}^{m+1}\right)^2 + \left(\Delta x_{j_-}^{m+1}\right)^2\right]. \end{aligned} \tag{3.15}$$

where $\Delta x_{j_-}^{m+1} = (x_j^{m+1} - x_{j-1}^{m+1})$ and $\Delta x_{j_+}^{m+1} = (x_{j+1}^{m+1} - x_j^{m+1})$. Thus, we achieve an expression for u_j^{m+1} on the non-uniform mesh by eliminating g_j^{m+1} from (3.14) and (3.15), as

$$u_j^{m+1} = \frac{\left(c_{j_+} + c_{j_-}\right)\left[\left(\Delta x_{j_+}^{m+1}\right)^2 + \left(\Delta x_{j_-}^{m+1}\right)^2\right] - \left(c_{j_-} - c_{j_+}\right)\left[\left(\Delta x_{j_+}^{m+1}\right)^2 - \left(\Delta x_{j_-}^{m+1}\right)^2\right]}{\left(x_{j+1}^{m+1} - x_{j-1}^{m+1}\right)\left[\left(\Delta x_{j_+}^{m+1}\right)^2 + \left(\Delta x_{j_-}^{m+1}\right)^2\right] - \left[\Delta x_{j_+}^{m+1} - \Delta x_{j_-}^{m+1}\right]\left[\left(\Delta x_{j_+}^{m+1}\right)^2 - \left(\Delta x_{j_-}^{m+1}\right)^2\right]},$$

which is exact for linear u_j. This reduces to

$$u_j^{m+1} = \frac{\left(\Delta x_{j_+}^{m+1}\right)^2 c_{j_-} + \left(\Delta x_{j_-}^{m+1}\right)^2 c_{j_+}}{\left(\Delta x_{j_+}^{m+1}\right)^2 \Delta x_{j_-}^{m+1} + \left(\Delta x_{j_-}^{m+1}\right)^2 \Delta x_{j_+}^{m+1}},$$

which can be written as

$$u_j^{m+1} = \frac{\frac{c_{j_-}/\Delta x_{j_-}^{m+1}}{\Delta x_{j_-}^{m+1}} + \frac{c_{j_+}/\Delta x_{j_+}^{m+1}}{\Delta x_{j_+}^{m+1}}}{\frac{1}{\Delta x_{j_-}^{m+1}} + \frac{1}{\Delta x_{j_+}^{m+1}}}, \tag{3.16}$$

an inversely weighted average of $\frac{c_{j_-}}{\Delta x_{j_-}^{m+1}}$ and $\frac{c_{j_+}}{\Delta x_{j_+}^{m+1}}$, $j = 1, \ldots, N-1$, which has the status of an interpolation formula. Furthermore, (3.16) simplifies to the mid-point rule (3.10) for an equally spaced mesh.

The full algorithm is

Given a mesh x_j^m and solution u_j^m, $j = 0, \ldots, N$ at time t^m, $m \geq 0$:

- Compute the mesh velocity v_j^m from a discrete form of (3.8). Determine the velocity at the boundaries from an appropriate extrapolation scheme;
- Compute the updated mesh x_j^{m+1} by a time-stepping scheme;
- Compute the updated solution u_j^{m+1} from (3.10) or (3.16). The solution at the boundaries, u_0^{m+1}, u_N^{m+1}, are given by the boundary conditions.

Examples are given later chapters.

We have shown how we can solve a problem that conserves mass using a moving mesh approach that conserves partial mass fractions. We now generalise this approach to solve problems that do not conserve mass.

3.2 A problem that does not conserve mass

For problems that do not conserve mass, we can adapt the moving mesh approach described in §3.1 so that we conserve *normalised* or partial mass fractions. For problems that conserve mass, the method given in this section reduces to the method given in §3.1.

Consider an IBVP that does not conserve mass, where the solution $u(x,t)$, $x \in (a(t), b(t))$, $t > t^0$ is positive in the interior and t^0 represents the initial time. The IBVP satisfies the PDE

$$\frac{\partial u}{\partial t} = \mathcal{G}u, \tag{3.17}$$

where again $\mathcal{G}$ is a purely spatial operator, with zero Dirichlet boundary conditions,

$$u = 0 \quad \text{at} \quad x = a(t), b(t), \quad t > t^0, \tag{3.18}$$

and initial condition

$$u(x, t^0) = u^0(x).$$

Remark 3.2.1 *As in §3.1, we again consider only IBVPs for which $u(x,t) > 0$, $x \in (a(t), b(t))$ holds for all $t \geq t^0$.*

The total mass of the solution at time t is

$$\theta(t) = \int_{a(t)}^{b(t)} u(x,t)\,\mathrm{d}x, \tag{3.19}$$

where the initial total mass $\theta(t^0)$ can be computed from the initial data.

We use the same mesh definition given in §3.1 where the time-dependent variable $\tilde{x}_j(t)$, $j = 0, \ldots, N$, represents the $N+1$ nodes of the mesh.

To determine the mesh velocity we adapt the partial mass conserving result (3.6) such that we conserve *normalised* partial masses,

$$c_j = \frac{1}{\theta(t)} \int_{a(t)}^{\tilde{x}_j(t)} u(x,t)\,\mathrm{d}x, \tag{3.20}$$

where c_j remains constant in time and $\theta(t)$ is the total area. Note that $c_N = 1$.

The procedure is as follows. Given a mesh $\tilde{x}_j(t)$ and solution $\tilde{u}_j(t) = u(\tilde{x}_j(t), t)$, we evaluate the total mass $\theta(t)$ directly from (3.19). To evaluate an updated value of the total mass (which is required for determining the updated solution) we first compute $\dot{\theta}(t)$. Then the mesh velocity $\tilde{v}_j(t)$ is computed. The mesh and total mass are updated simultaneously using a time-stepping scheme. This enables us to recover the updated solution on the new mesh. Details are given in the following subsections.

3.2.1 Determining the rate of change of total mass

The total mass $\theta(t)$ can be obtained by differentiating (3.19) with respect to time using the Leibnitz integral rule, giving

$$\dot{\theta}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} u(x,t)\,\mathrm{d}x = \int_{a(t)}^{b(t)} \frac{\partial u}{\partial t}\,\mathrm{d}x + \tilde{u}_N(t)\tilde{v}_N(t) - \tilde{u}_0(t)\tilde{v}_0(t).$$

Substituting $\frac{\partial u}{\partial t}$ from (3.17), and using the boundary conditions (3.18),

$$\dot{\theta}(t) = \int_{a(t)}^{b(t)} \mathcal{G}u\,\mathrm{d}x. \tag{3.21}$$

For a specific $\mathcal{G}$, the integral in (3.21) can be approximated using quadrature to give a discrete form of (3.21).

3.2.2 Determining the mesh velocity

To find an expression for the mesh velocity $\tilde{v}_j(t)$, we differentiate (3.20) with respect to time using the Leibnitz integral rule, giving

$$\dot{\theta}(t)c_j = \frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{\tilde{x}_j(t)} u(x,t)\,\mathrm{d}x = \int_{a(t)}^{\tilde{x}_j(t)} \frac{\partial u}{\partial t}\,\mathrm{d}x + \tilde{u}_j(t)\tilde{v}_j(t) - \tilde{u}_0(t)\tilde{v}_0(t).$$

Substituting $\frac{\partial u}{\partial t}$ from the PDE (3.17), and using the boundary conditions (3.18),

$$\dot{\theta}(t)c_j = \int_{a(t)}^{\tilde{x}_j(t)} \mathcal{G}u\,\mathrm{d}x + \tilde{u}_j(t)\tilde{v}_j(t). \tag{3.22}$$

Thus, for $\tilde{u}_j(t) \neq 0$, the nodes move such that

$$\tilde{v}_j(t) = \frac{1}{\tilde{u}_j(t)}\left(\dot{\theta}(t)c_j - \int_{a(t)}^{\tilde{x}_j(t)} \mathcal{G}u\,\mathrm{d}x\right), \tag{3.23}$$

where $\dot{\theta}$ is given by (3.21). Recalling Remark 3.2.1, equation (3.23) holds for interior nodes $j = 1, \ldots, N-1$. Again, the mesh velocities at the boundaries, $\tilde{v}_0(t), \tilde{v}_N(t)$, can be extrapolated from the internal mesh velocities. We observe that for constant total mass, equation (3.23) is equivalent to (3.8).

As in §3.2.1, for a specific $\mathcal{G}$ the integral in (3.23) may be approximated using quadrature to give a discrete form of (3.23).

3.2.3 Advancing the total mass and mesh in time

We choose Δt and $t^m = m\Delta t$, $\Delta t > 0$, $m = 0, 1, \ldots$ as before, and repeat the notation of §3.1, with the additions $\theta^m \approx \theta(t^m)$ and $\dot{\theta}^m \approx \dot{\theta}(t^m)$. For a given total mass θ^m, mesh x_j^m, and solution u_j^m, $j = 0, \ldots, N$, we compute the rate of change of total mass $\dot{\theta}^m$ and the mesh velocity v_j^m, and use a time-stepping scheme to update the total mass θ^{m+1} and mesh x_j^{m+1}.

3.2.4 Recovering the solution

To approximate the updated solution u_j^{m+1}, we equate (3.20) between the points $\tilde{x}_{j+1}$ and $\tilde{x}_{j-1}$, at time-levels $t = t^{m+1}$ and $t = t^0$, giving

$$c_{j+1} - c_{j-1} = \frac{1}{\theta(t^{m+1})} \int_{x_{j-1}^{m+1}}^{x_{j+1}^{m+1}} u(x, t^{m+1}) \, \mathrm{d}x = \frac{1}{\theta(t^0)} \int_{x_{j-1}^{0}}^{x_{j+1}^{0}} u(x, t^0) \, \mathrm{d}x, \tag{3.24}$$

for $j = 1, \cdots, N-1$. We note that for constant total mass equation (3.24) is equivalent to (3.9). As in §3.1.3 we use two different quadrature rules to evaluate the integrals in (3.24) so that we can recover u_j^{m+1}.

The first order mid-point approximation gives

$$\frac{1}{\theta^{m+1}} \left(x_{j+1}^{m+1} - x_{j-1}^{m+1} \right) u_j^{m+1} = \frac{1}{\theta^0} \left(x_{j+1}^0 - x_{j-1}^0 \right) u_j^0.$$

Hence, the updated solution can be determined by

$$u_j^{m+1} = \frac{\theta^{m+1}}{\theta^0} \frac{\left(x_{j+1}^0 - x_{j-1}^0 \right)}{\left(x_{j+1}^{m+1} - x_{j-1}^{m+1} \right)} u_j^0, \tag{3.25}$$

for $j = 1, \cdots, N-1$. As in §3.1.3, the solution at the boundaries, u_0^{m+1}, u_N^{m+1}, is determined by the boundary conditions (3.18). Note that at any given time-level the discrete *relative* partial mass $\frac{1}{\theta^{m+1}}(x_{j+1}^{m+1} - x_{j-1}^{m+1})u_j^{m+1}$ is preserved, and is equal to the corresponding initial *relative* partial mass $\frac{1}{\theta^0}(x_{j+1}^0 - x_{j-1}^0)u_j^0$.

Equation (3.25) is the same as equation (3.10), with an additional ratio of the total masses at time-levels $t = t^0$ and $t = t^{m+1}$. Therefore, a second order method to recover

u_j^{m+1} (which is exact for a linear solution on a non-uniform mesh) is the same as (3.16), but with the addition of the ratio of total masses $\frac{\theta^{m+1}}{\theta^0}$, i.e.

$$u_j^{m+1} = \frac{\theta^{m+1}}{\theta^0} \frac{\frac{c_{j_-}/\Delta x_{j_-}^{m+1}}{\Delta x_{j_-}^{m+1}} + \frac{c_{j_+}/\Delta x_{j_+}^{m+1}}{\Delta x_{j_+}^{m+1}}}{\frac{1}{\Delta x_{j_-}^{m+1}} + \frac{1}{\Delta x_{j_+}^{m+1}}}, \tag{3.26}$$

for $j = 1, \ldots, N-1$.

The full algorithm

Given a mesh x_j^m, solution u_j^m and total mass θ^m, $j = 0, \ldots, N$,at time t^m, $m \geq 0$:

- Compute $\dot{\theta}^m$ from a discrete form of (3.21);
- Compute the mesh velocity v_j^m from a discrete form of (3.23). Determine the velocity at the boundaries from an appropriate extrapolation scheme;
- Compute the updated total mass θ^{m+1} and mesh x_j^{m+1} by a time-stepping scheme;
- Compute the updated solution u_j^{m+1} from (3.25) or (3.26). The solution at the boundaries, u_0^{m+1}, u_N^{m+1}, are given by the boundary conditions.

Examples are given in later chapters.

We have shown how we can solve a problem that does not conserves mass using a moving mesh approach that conserves *relative* partial mass fractions. We note that for a constant mass, the calculations given here reduce to the work presented in §3.1. We now consider an approach where we balance the partial masses with the source term.

3.3 A method that preserves mass balance

Consider now an IVBP that does not conserve mass, of the particular form

$$\frac{\partial u}{\partial t} = \mathcal{H}u + S(x,t), \tag{3.27}$$

where $\mathcal{H}$ is a spatial operator, and S is a source term, such that mass is conserved when $S = 0$. We consider zero Dirichlet boundary conditions,

$$u = 0 \quad \text{at} \quad x = a(t), b(t), \quad t > t^0, \tag{3.28}$$

and initial condition

$$u(x,t^0) = u^0(x),$$

where t^0 represents the initial time.

Remark 3.3.1 *As in §3.1, we again consider only IBVPs for which $u(x,t) > 0$, $x \in (a(t), b(t))$ holds for all $t \geq t^0$.*

In §3.2 we developed a moving mesh method by conserving normalised partial masses, which led to equation (3.20). In this section we develop a similar moving mesh method by assuming the partial mass balance,

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{a(t)}^{\tilde{x}_j(t)} u(x,t)\,\mathrm{d}x = \int_{a(t)}^{\tilde{x}_j(t)} S(x,t)\,\mathrm{d}x. \tag{3.29}$$

For this moving mesh scheme, we once more use the same notation given in §3.1 where the time dependent variables $\tilde{x}_j(t)$, $j = 0, \ldots, N$, represent the $N+1$ nodes of the mesh, which coincide instantaneously with x_j.

When using this method we define the *partial* masses $\Theta_j(t^m)$ at time t and interval points $\tilde{x}_j(t)$, as

$$\Theta_j(t) = \int_{a(t)}^{\tilde{x}_j(t)} u(x,t)\,\mathrm{d}x, \tag{3.30}$$

where $j = 0, \ldots, N$. For a problem that conserves mass, equation (3.30) is equivalent to (3.6) where $\Theta_j(t) = c_j$. The values $\Theta_j(t^0)$ can be computed from the initial data at $t = t^0$. Note that (3.30) varies in time, whereas the corresponding equation (3.20) defined c_j which were constant in time.

The procedure is as follows. Given a mesh $\tilde{x}_j(t)$ and solution $\tilde{u}_j(t) = u(\tilde{x}_j(t), t)$, we first evaluate the partial masses $\Theta_j(t)$ directly from (3.30). To evaluate updated values of the partial masses (which are required for determining the updated solution) we compute $\dot{\Theta}_j(t)$. The mesh velocity $\tilde{v}_j(t)$ is then computed, and the mesh and partial masses are updated simultaneously by a time-stepping scheme. This enables us to recover the updated solution on the new mesh. Details are given in the following subsections.

3.3.1 Determining the rate of change of partial masses

Given the solution $\tilde{u}_j(t)$ at the initial time we evaluate the partial masses $\Theta_j(t)$ directly from (3.30) using quadrature. To evaluate the updated partial masses we compute $\dot{\Theta}_j(t)$.

Substituting (3.30) into (3.29), we have

$$\dot{\Theta}_j(t) = \int_{a(t)}^{\tilde{x}_j(t)} S(x,t)\,\mathrm{d}x. \tag{3.31}$$

For a specific IBVP, the integral in (3.31) can then be approximated using quadrature to determine a discrete form of (3.31).

3.3.2 Determining the mesh velocity

To obtain an expression for the mesh velocity $\tilde{v}_j(t)$ we differentiate (3.30) with respect to time using the Leibnitz integral rule again, to give

$$\dot{\Theta}_j(t) = \frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{\tilde{x}_j(t)} u(x,t)\,\mathrm{d}x \quad = \quad \int_{a(t)}^{\tilde{x}_j(t)} \frac{\partial u}{\partial t}\,\mathrm{d}x + \tilde{u}_j(t)\tilde{v}_j(t) - \tilde{u}_0(t)\tilde{v}_0(t).$$

Substituting $\frac{\partial u}{\partial t}$ from (3.27), and using the boundary conditions (3.28),

$$\dot{\Theta}_j(t) \quad = \quad \int_{a(t)}^{\tilde{x}_j(t)} \{\mathcal{H}u + S(x,t)\}\,\mathrm{d}x + \tilde{u}_j(t)\tilde{v}_j(t). \tag{3.32}$$

Equating (3.32) and (3.31),

$$\int_{a(t)}^{\tilde{x}_j(t)} \mathcal{H}u\,\mathrm{d}x + \tilde{u}_j(t)\tilde{v}_j(t) = 0.$$

Thus, for $\tilde{u}_j(t) \neq 0$, the nodes move such that

$$\tilde{v}_j(t) \quad = \quad -\frac{1}{\tilde{u}_j(t)}\int_{a(t)}^{\tilde{x}_j(t)} \mathcal{H}u\,\mathrm{d}x. \tag{3.33}$$

Recalling Remark 3.3.1, equation (3.33) holds for interior points $j = 1, \ldots, N-1$. Again, the mesh velocities at the boundaries, $\tilde{v}_0(t), \tilde{v}_N(t)$, can be extrapolated from the interior mesh velocities. We observe that if there were no source term, equation (3.33) would be equivalent to (3.8) for $j = 1, 2, ..., N-1$.

Once more, as in §3.2.1, for a specific IBVP the integral in (3.33) is approximated using quadrature to give a discrete form of (3.33).

3.3.3 Advancing the partial masses and mesh in time

We choose Δt and $t^m = m\Delta t$, $\Delta t > 0$, $m = 0, 1, \ldots$ as before, and repeat the notation of §3.1, with the additions $\Theta_j^m \approx \Theta_j(t^m)$ and $\dot{\Theta}_j^m \approx \dot{\Theta}_j(t^m)$. For given partial masses Θ_j^m, mesh x_j^m, and solution u_j^m, $j = 0, \ldots, N$, we compute the rate of change of partial masses $\dot{\Theta}_j^m$ and the mesh velocity v_j^m, and use a time-stepping scheme to update the partial masses Θ_j^{m+1} and mesh x_j^{m+1}.

3.3.4 Recovering the solution

Once approximations to the updated partial masses and the mesh velocity have been determined, the final step is to recover the solution. We first use the relation (3.30) to determine the difference between $\Theta_{j+1}(t)$ and $\Theta_{j-1}(t)$,

$$\Theta_{j+1}(t) - \Theta_{j-1}(t) = \int_{\tilde{x}_{j-1}(t)}^{\tilde{x}_{j+1}(t)} u(x,t)\,\mathrm{d}x.$$

Using a first order mid-point approximation to evaluate the integral at t^{m+1} gives

$$\Theta_{j+1}^{m+1} - \Theta_{j-1}^{m+1} = (x_{j+1}^{m+1} - x_{j-1}^{m+1})u_j^{m+1}.$$

Hence, the updated solution $\tilde{u}_j(t^{m+1})$ can be determined by

$$u_j^{m+1} = \frac{\Theta_{j+1}^{m+1} - \Theta_{j-1}^{m+1}}{x_{j+1}^{m+1} - x_{j-1}^{m+1}}, \tag{3.34}$$

for $j = 1, \ldots, N-1$. The solution at the boundaries, u_0^{m+1}, u_N^{m+1}, is determined by the boundary conditions (3.2).

We can also derive an expression for recovering u_j^{m+1}, which is exact for linear u_j on an irregular mesh by applying the same principles used to derive (3.16). For this method we replace c_{j_+} by $\left(\Theta_{j+1}^{m+1} - \Theta_j^{m+1}\right)$ and c_{j_-} by $\left(\Theta_j^{m+1} - \Theta_{j-1}^{m+1}\right)$ since u_j^{m+1} is recovered using the updated partial masses, not the initial partial masses. This gives

$$u_j^{m+1} = \frac{\frac{\left(\Theta_j^{m+1} - \Theta_{j-1}^{m+1}\right)/\Delta x_{j_-}^{m+1}}{\Delta x_{j_-}^{m+1}} + \frac{\left(\Theta_{j+1}^{m+1} - \Theta_j^{m+1}\right)/\Delta x_{j_+}^{m+1}}{\Delta x_{j_+}^{m+1}}}{\frac{1}{\Delta x_{j_-}^{m+1}} + \frac{1}{\Delta x_{j_+}^{m+1}}}, \tag{3.35}$$

for $j = 1, \ldots, N-1$. This expression is the same as (3.16) for constant partial masses.

The full algorithm

Given a mesh x_j^m, solution u_j^m and partial masses Θ_j^m, $j = 0, \ldots, N$, at time t^m, $m \geq 0$:

- Compute the partial masses $\dot{\Theta}_j^m$ from a discrete form of (3.31);

- Compute the mesh velocity v_j^m from a discrete form of (3.33). Determine the velocity at the boundaries from an appropriate extrapolation scheme;
- Compute the updated partial masses Θ_j^{m+1} and mesh x_j^{m+1};
- Compute the updated solution u_j^{m+1} from (3.34) or (3.35). The solution at the boundaries, u_0^{m+1}, u_N^{m+1}, are given by the boundary conditions.

Examples are discuses in Chapters 6 and 7.

We have shown how we can solve a problem that does not conserves mass using a moving mesh approach that balances the partial mass fractions with a source term. We now look at some of the time-stepping schemes that we use with our moving mesh method.

3.4 Time-stepping schemes

When implementing the three moving mesh schemes given in §3.1–3.3 we use various time-stepping schemes to update the mesh velocity $\tilde{x}_j(t)$, total mass $\theta(t)$, and partial masses $\Theta_j(t)$.

3.4.1 Explicit schemes

The simplest time-stepping method we use is the first order explicit Euler time-stepping scheme, giving

$$\frac{x_j^{m+1} - x_j^m}{\Delta t} = v_j^m \quad \text{and hence} \quad x_j^{m+1} = x_j^m + \Delta t\, v_j^m, \tag{3.36}$$

$$\frac{\theta^{m+1} - \theta^m}{\Delta t} = \dot{\theta}^m \quad \text{and hence} \quad \theta^{m+1} = \theta^m + \Delta t\, \dot{\theta}^m, \tag{3.37}$$

$$\frac{\Theta_j^{m+1} - \Theta_j^m}{\Delta t} = \dot{\Theta}_j^m \quad \text{and hence} \quad \Theta_j^{m+1} = \Theta_j^m + \Delta t\, \dot{\Theta}_j^m. \tag{3.38}$$

The explicit Euler time-stepping scheme requires small Δt so that $\tilde{x}_j$ remains monotonic in j, i.e. the x_j^m do not 'tangle' so (3.5) holds, see Figure 3.1.

We also considered Runge-Kutta explicit time-stepping schemes. Adaptive predictor-correcter Runge-Kutta time-stepping allows greater accuracy and optimal time steps without losing stability. The time step is chosen to be as large as possible whilst ensuring that the difference between the solutions from two explicit Runge-Kutta formulas conforms to a certain tolerance. We implemented both ODE23 and ODE45 in Matlab on our examples.

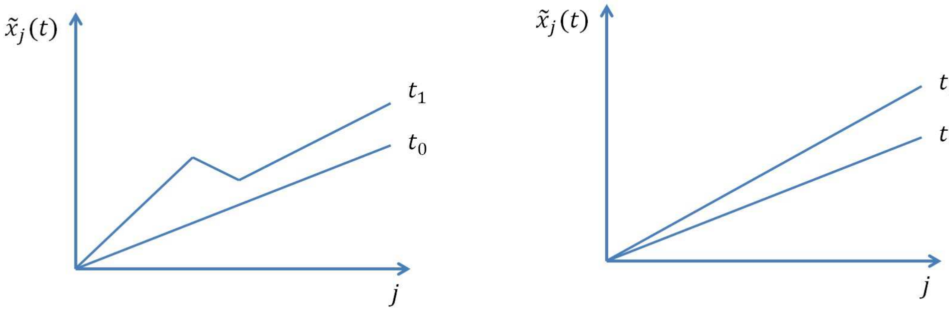


Fig. 3.1: Diagrams to illustrate the relation between $\tilde{x}_j(t)$ and j for a mesh that is tangled compared to one that is not tangled. The graph on the left shows a mesh that is tangled (not monotonic) at time t_1.

Both solvers invoke explicit Runge-Kutta methods to integrate the system of $N-1$ ordinary differential equations. The solver ODE23 uses second and third order Runge-Kutta formulas, and ODE45 uses fourth and fifth order Runge-Kutta formulas, which are more restricting on the size of the time-step, corresponding to a more accurate solution.

A third solver is ODE15s, which is designed to solve a stiff system using implicit time-stepping. Implementing this solver on our examples produced results similar to results obtained with both ODE23 and ODE45 (which are not designed for stiff systems), inferring that the method is not particularly stiff in these cases.

3.4.2 A semi-implicit scheme

When using the mass conserving approach in §3.1, we used a semi-implicit time-stepping scheme to update the internal nodes of the mesh $\tilde{x}_j(t)$, $j = 1, \ldots, N-1$. A semi-implicit scheme remains stable for large Δt. To ensure mesh tangling does not occur we develop a semi-implicit scheme that depends on the mesh velocity $\tilde{v}_j(t)$ determined for each problem. The general structure of our semi-implicit approach is to first discretise the mesh velocity such that

$$v_j^m = \frac{1}{\Delta x_j^m}\left(\phi_{j+}^m - \phi_{j-}^m\right) \qquad j = 1, \ldots, N-1,$$

where $\Delta x_j^m = x_{j+\frac{1}{2}}^m - x_{j-\frac{1}{2}}^m$, and $\phi_{j\pm}^m$ refers to left and right intervals of the mesh velocity equation (3.8) integrated with respect to x, namely

$$\phi_{j+}^m = \int_{a(t)}^{\tilde{x}_{j+\frac{1}{2}}(t)} \tilde{v}_j(t)\,\mathrm{d}x \qquad \text{and} \qquad \phi_{j-}^m = \int_{a(t)}^{\tilde{x}_{j-\frac{1}{2}}(t)} \tilde{v}_j(t)\,\mathrm{d}x.$$

Then, assuming that the mesh $\tilde{x}_j(t)$ changes smoothly in time, we alter the Euler scheme (3.36) to be semi-implicit, in the manner

$$\frac{x_j^{m+1} - x_j^m}{\Delta t} \quad = \quad \frac{1}{\Delta x_j^m}\left(\phi_{j+}^m \frac{\Delta x_{j\pm}^{m+1}}{\Delta x_{j\pm}^m} - \phi_{j-}^m \frac{\Delta x_{j\pm}^{m+1}}{\Delta x_{j\pm}^m}\right), \quad j = 1, \ldots, N-1, \quad (3.39)$$

where $\Delta x_{j+}^m = (x_{j+1}^m - x_j^m)$ and $\Delta x_{j-}^m = (x_j^m - x_{j-1}^m)$. We choose $\Delta x_{j\pm}^{m+1}$ to be either Δx_{j+}^{m+1} or Δx_{j-}^{m+1}, that is

$$\frac{x_j^{m+1} - x_j^m}{\Delta t} = \begin{cases} \frac{1}{\Delta x_j^m}\left(\phi_{j+}^m \frac{\Delta x_{j+}^{m+1}}{\Delta x_{j+}^m} - \phi_{j-}^m \frac{\Delta x_{j-}^{m+1}}{\Delta x_{j-}^m}\right) & \text{for} \quad \phi_{j+}^m, \phi_{j-}^m > 0; \\ \frac{1}{\Delta x_j^m}\left(\phi_{j+}^m \frac{\Delta x_{j+}^{m+1}}{\Delta x_{j+}^m} - \phi_{j-}^m \frac{\Delta x_{j+}^{m+1}}{\Delta x_{j+}^m}\right) & \text{for} \quad \phi_{j+}^m > 0, \phi_{j-}^m < 0; \\ \frac{1}{\Delta x_j^m}\left(\phi_{j+}^m \frac{\Delta x_{j-}^{m+1}}{\Delta x_{j-}^m} - \phi_{j-}^m \frac{\Delta x_{j-}^{m+1}}{\Delta x_{j-}^m}\right) & \text{for} \quad \phi_{j+}^m, \phi_{j-}^m < 0; \\ \frac{1}{\Delta x_j^m}\left(\phi_{j+}^m \frac{\Delta x_{j-}^{m+1}}{\Delta x_{j-}^m} - \phi_{j-}^m \frac{\Delta x_{j+}^{m+1}}{\Delta x_{j+}^m}\right) & \text{for} \quad \phi_{j+}^m < 0, \phi_{j-}^m > 0, \end{cases} \quad (3.40)$$

for reasons that will be explained below. The boundary values, $\tilde{x}_0(t), \tilde{x}_N(t)$, are updated explicitly by a first order scheme. These are calculated before the internal nodes so that $\Delta x_{1_-}^{m+1}$ and $\Delta x_{N-1_+}^{m+1}$ can be determined. The whole scheme is then first order in time.

Theorem 3.4.1 *The semi-implicit scheme (3.39) ensures that the mesh does not tangle, i.e.*

$$x_{j-1}^m < x_j^m < x_{j+1}^m, \quad (3.41)$$

for all $j = 0, 1, \ldots, N$ and all time t^m, $m = 1, 2, \ldots$, provided the $\Delta x_{j\pm}^{m+1}$ are chosen to be either Δx_{j+}^{m+1} or Δx_{j-}^{m+1} according to the four parts of (3.40).

Proof Given that the mesh is not tangled at time-level t^m we show that (3.41) holds for all subsequent time-levels by proving that the maximum and minimum of the set $\{x_j^{m+1}\}$ occur at the boundaries, for all $j = 1, \ldots, N-1$.

We first prove the maximum principle, by contradiction. Suppose that an isolated maximum of x occurs at the interior point x_j^{m+1}. We consider the sign of each term in (3.39), and subsequently determine that the sign of right-hand side contradicts that of the left-hand side. Since (3.41) holds at time-level t^m we have that

$$\Delta x_j^m, \Delta x_{j\pm}^m > 0,$$

Now, if the interior point x_j^{m+1} is a maximum then

$$\begin{aligned} x_j^{m+1} - x_j^m &> 0, && (3.42) \\ \Delta x_{j-}^{m+1} = x_j^{m+1} - x_{j-1}^{m+1} &> 0, && (3.43) \\ \Delta x_{j+}^{m+1} = x_{j+1}^{m+1} - x_j^{m+1} &< 0. && (3.44) \end{aligned}$$

The inequality (3.42) implies that the left-hand side of (3.39) is positive. To complete the proof by contradiction, we note that $\Delta x_{j\pm}^{m+1}$ gives a negative right-hand side if $\Delta x_{j\pm}^{m+1}$ is not carefully considered. For example, suppose $\phi_{j+}^m, \phi_{j-}^m < 0$ and the $\Delta x_{j\pm}^{m+1}$ of (3.39) are determined such that

$$\frac{x_j^{m+1} - x_j^m}{\Delta t} = \frac{1}{\Delta x_j^m}\left(\phi_{j+}^m \frac{\Delta x_{j-}^{m+1}}{\Delta x_{j-}^m} - \phi_{j-}^m \frac{\Delta x_{j+}^{m+1}}{\Delta x_{j+}^m}\right),$$

This gives a negative right-hand side, contradicting the left-hand side. We have therefore shown that an isolated maximum cannot occur. Using the same reasoning given here, it can be shown that a set of equal maximum values cannot occur. Hence the maximum occurs at the boundary.

A minimum principle can be proved with the same reasoning to show that an isolated minimum can not occur at any interior point when $\Delta x_{j\pm}^{m+1}$ is specified according to (3.40). The proof shows that x_j^m, $j = 0, 1, ..., N$, is bounded by its neighbours. It also extends to non-isolated interior points and hence, equation (3.39) with appropriate $\Delta x_{j\pm}^{m+1}$ (determined by (3.40)), ensures that the mesh is monotonic in space and is bounded by its endpoint values, so that tangling cannot occur.

Examples are given in the Chapters which follow. □

We now turn to some specific problems to illustrate the working of the finite difference moving mesh method. We begin with the PME.

4

The Porous Medium Equation

4.1 Introduction

The Porous Medium Equation (PME)

$$\frac{\partial u}{\partial t} = \nabla \left(u(x,t)^n \nabla u \right), \tag{4.1}$$

where $n \geq 1$ is one of the simplest nonlinear evolution equations of parabolic type. It can be used to describe physical situations such as fluid flow, heat transfer or diffusion. Most notably, it is used to describe the flow of a perfect gas in a homogeneous porous medium.

The main aim of this chapter is to solve the PME numerically using the moving mesh method described in §3.1.

We begin this chapter by deriving the PME from a general form of Darcy's Law in §4.2, as shown in [99]. Then, before solving the PME numerically, we discuss some of the properties of the PME which our numerical scheme aims to preserve, in §4.3. One of these properties is self-similarity, so in §4.4 we derive a self-similar solution, as originally presented in [12]. This self-similar solution is used to provide the initial conditions when we solve the PME numerically, and also to compare the numerical solution to an exact solution in the results section, §4.7. Finally, in §4.8 we present results using the moving mesh finite element method discussed in §2.3.

4.2 Deriving the PME from Darcy's Law

We first derive the PME by considering three model equations which relate variables associated with gas flow through a porous medium.

(i) Mass balance

We assume that the flow of gas obeys the equation of continuity

$$\epsilon \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = \mathbf{0}, \tag{4.2}$$

where $\epsilon \in (0,1)$ is the porosity of the medium, ρ is the density, and $\mathbf{V}$ is the velocity. Here $\nabla \cdot$ represents the divergence operator.

(ii) Darcy's Law

Darcy's Law was formulated by the French engineer H. Darcy in 1856 [99]. It models the flow of a fluid (or gas) through a porous medium. Maintaining the notation in (4.2),

$$\mu \mathbf{V} = -\kappa \nabla p, \tag{4.3}$$

where ∇p is the pressure gradient vector, κ is the permeability tensor (assumed to be a strictly positive constant in most applications), and $\mu > 0$ is the viscosity of the fluid (or gas).

(iii) Equation of state

The equation of state for a perfect gas is

$$p = p_0 \rho^{\gamma}, \tag{4.4}$$

where p is the pressure, p_0 is the reference pressure and $\gamma \geq 1$ is the ratio of specific heats for the gas.

Substituting Darcy's Law (4.3) and the equation of state (4.4) into the mass conservation equation (4.2) gives

$$\frac{\partial \rho}{\partial t} = \frac{\kappa p_0}{\epsilon \mu} \nabla.(\rho \nabla \rho^{\gamma}). \tag{4.5}$$

Now

$$\rho \nabla \rho^{\gamma} \;\; = \;\; \rho \gamma \rho^{\gamma - 1} \nabla \rho = \gamma \rho^{\gamma} \nabla \rho,$$

so we may write (4.5) as

$$\frac{\partial \rho}{\partial t} = \frac{\gamma \kappa p_0}{\epsilon \mu} \nabla.(\rho^{\gamma} \nabla \rho).$$

The constant $\frac{\gamma \kappa p_0}{\epsilon \mu}$ can be scaled out (define for instance a new time, $t' = \frac{\gamma \kappa p_0}{\epsilon \mu} t$), thus leaving us with the PME. We adapt this result to meet standard notation by writing $\gamma = n$, and $\rho = u$, giving

$$\frac{\partial u}{\partial t} = \nabla \cdot (u^n \nabla u),$$

where $n \geq 1$. In one-dimension this equation is

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x} \left(u^n \frac{\partial u}{\partial x} \right). \tag{4.6}$$

More generally, for the radially symmetric case,

$$\frac{\partial u}{\partial t} = \frac{1}{r^{\delta - 1}} \frac{\partial}{\partial r} \left(r^{\delta - 1} u^n \frac{\partial u}{\partial r} \right), \tag{4.7}$$

where r is the radial coordinate and $\delta = 1, 2, 3$ is the dimension. When $\delta = 1$, we have the one-dimensional case (which is denoted with Cartesian coordinates throughout this thesis), while $\delta = 2$ describes the radially symmetric two-dimensional case.

The PME is parabolic everywhere except where $u = 0$, where the spatial operator is degenerate. This prevents the derivation of a strong solution at these points, because there may exist an interface or free boundary separating regions where $u > 0$ from regions where $u = 0$. Therefore theoretical work focuses on weak solutions for the PME. The PME has been well documented in [99], where several chapters are devoted to weak solutions of the PME problem, with a further chapter devoted to self-similar solutions of the PME. In this chapter we derive such a self-similar solution. However, firstly we discuss some properties of the PME.

4.3 Properties of the PME in one dimension

We are interested in several properties of the PME for the one-dimensional Cartesian case (4.6) with its boundary defined by the edge of the support, i.e.

$$u = 0 \quad \text{at} \quad x = a(t), b(t), \quad t > 0. \tag{4.8}$$

In [99] many properties of the PME are given, and proved. We prove two well-established properties which our numerical approach relies upon: conservation of mass, and stationary centre of mass.

Lemma 4.3.1 *The one-dimensional PME conserves mass in time.*

Proof To prove that the total mass does not change over time we show that the derivative of the total mass (in time) is zero. Using the Leibnitz integral rule,

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} u(x,t)\,\mathrm{d}x \;=\; \int_{a(t)}^{b(t)} \frac{\partial u}{\partial t}\,\mathrm{d}x + u(b,t)\frac{\mathrm{d}b}{\mathrm{d}t} - u(a,t)\frac{\mathrm{d}a}{\mathrm{d}t},$$

Substituting $\frac{\partial u}{\partial t}$ from (4.6), and setting the last two terms to zero due to the boundary conditions (4.8),

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} u(x,t)\,\mathrm{d}x &= \int_{a(t)}^{b(t)} \frac{\partial}{\partial x}\left(u(x,t)^n \frac{\partial u}{\partial x}\right)\,\mathrm{d}x,\\ &= u(b,t)^n\frac{\partial b}{\partial x} - u(a,t)^n\frac{\partial a}{\partial x}.\end{aligned}$$

The right-hand side is zero, due to boundary conditions (4.8) again, hence

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} u(x,t)\,\mathrm{d}x \;=\; 0,$$

as required. The result is easily extended to the radially symmetric case

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{0}^{R(t)} u(r,t)r^{\delta-1}\,\mathrm{d}r = 0,$$

where $R(t)$ is the radius and $\delta = 2, 3$ is the number of dimensions. □

Lemma 4.3.2 *The one-dimensional PME has a stationary centre of mass.*

Proof The centre of mass $\bar{x}(t)$ in one dimension is defined by the ratio

$$\bar{x}(t) = \frac{\int_{a(t)}^{b(t)} u(x,t)x \, \mathrm{d}x}{\int_{a(t)}^{b(t)} u(x,t) \, \mathrm{d}x}. \tag{4.9}$$

To demonstrate that the centre of mass $\bar{x}(t)$ does not move, it is sufficient to show that

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} u(x,t)x \, \mathrm{d}x = 0,$$

since the denominator of (4.9) is constant in time. By the Leibnitz integral rule,

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} u(x,t)x \, \mathrm{d}x \quad = \quad \int_{a(t)}^{b(t)} x\frac{\partial u}{\partial t} \, \mathrm{d}x + u(b,t)b(t) \, \frac{\mathrm{d}b}{\mathrm{d}t} - u(a,t)a(t) \, \frac{\mathrm{d}a}{\mathrm{d}t}.$$

Substituting $\frac{\partial u}{\partial t}$ from the PME (4.6), and eliminating the last two terms from the boundary conditions (4.8), gives

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} u(x,t)x \, \mathrm{d}x \quad = \quad \int_{a(t)}^{b(t)} x\frac{\partial}{\partial x}\left(u(x,t)^n\frac{\partial u}{\partial x}\right) \, \mathrm{d}x.$$

Using integration by parts on the right-hand side,

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} u(x,t)x \, \mathrm{d}x \quad = \quad b(t)u(b,t)^n\frac{\partial b}{\partial x} - a(t)u(a,t)^n\frac{\partial a}{\partial x} - \int_{a(t)}^{b(t)} u(x,t)^n\frac{\partial u}{\partial x} \, \mathrm{d}x.$$

The first two terms on the right-hand side vanish again due to the boundary conditions (4.8). The final term on the right-hand side is then rearranged to give

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} u(x,t)x \, \mathrm{d}x \quad &= \quad -\frac{1}{n+1}\int_{a(t)}^{b(t)} \frac{\partial}{\partial x}\left(u(x,t)^{n+1}\right) \, \mathrm{d}x, \\ &= \quad -\frac{1}{n+1}\left(u(b,t)^{n+1} - u(a,t)^{n+1}\right),\end{aligned}$$

which again vanishes due to the boundary conditions (4.8). Hence, we achieve the required result

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} u(x,t)x \, \mathrm{d}x \quad = \quad 0. \tag{4.10}$$

In the radially symmetric case the origin is fixed. □

By showing that the mass is conserved, we know that updating the total mass at each time-level is not required when applying our moving mesh method. In addition, knowing that the centre of mass remains stationary allows us to set the mesh velocity to zero at that point. Before numerically solving the PME, we recall the self-similar solution to the PME in the next section.

4.4 A self-similar solution

In this section we recall a class of exact solutions to the radially symmetric PME (4.7) that are invariant under a scaling group in the variables (t, r, u), and therefore take the so-called self-similar form.

A time dependent phenomenon is called self-similar if the spatial distributions of its variables at different times can be obtained from one another by a similarity transform [11], which is a transformation that maintains certain features of a function or curve. A particular similarity transformation is a scale-invariant transformation, where the variables are scaled by powers of a common factor λ.

A prerequisite to deriving a self-similar scaling solution is determining a scale-invariant transformation of the PME [11].

4.4.1 Scale invariance

Scale invariance, defined as the invariance of the PDE under scaling, originated from the analysis of the consequences of the change of units of measurement on the mathematical form of the laws of physics [11]. Mathematically, it can be viewed as a particular aspect of the study of the invariance of differential equations under general groups of transformations.

Consider the scaling transformation

$$t = \lambda \hat{t}, \qquad r = \lambda^{\beta} \hat{r}, \qquad u = \lambda^{\gamma} \hat{u}, \tag{4.11}$$

where λ is the scaling parameter and β and γ are constants. (The γ here is distinct from the ratio of specific heats γ in §4.2.)

Transforming the derivatives in the PME into the variables $\hat{t}$, $\hat{r}$ and $\hat{u}$ gives

$$\frac{\partial u}{\partial t} = \lambda^{\gamma-1} \frac{\partial \hat{u}}{\partial \hat{t}}, \tag{4.12}$$

$$\frac{\partial u}{\partial r} = \lambda^{\gamma-\beta} \frac{\partial \hat{u}}{\partial \hat{r}}. \tag{4.13}$$

The first of these (4.12) is the left-hand side of the PME. Using (4.13) we transform the right-hand side to get

$$\frac{1}{r^{\delta-1}}\frac{\partial}{\partial r}\left(r^{\delta-1}u(r,t)^n\frac{\partial u}{\partial r}\right) \quad = \quad \frac{\lambda^{\gamma(n+1)-2\beta}}{\hat{r}^{\delta-1}}\frac{\partial}{\partial \hat{r}}\left(\hat{r}^{\delta-1}\hat{u}(\hat{r},\hat{t})^n\frac{\partial \hat{u}}{\partial \hat{r}}\right).$$

Hence our transformed general PME is

$$\lambda^{\gamma-1}\frac{\partial \hat{u}}{\partial \hat{t}} = \frac{\lambda^{\gamma(n+1)-2\beta}}{r^{\delta-1}}\frac{\partial}{\partial \hat{r}}\left(\hat{r}^{\delta-1}\hat{u}(\hat{r},\hat{t})^n\frac{\partial \hat{u}}{\partial \hat{r}}\right).$$

Therefore, for the PME (4.7) to be invariant under the transformation (4.11) we require

$$\gamma - 1 = \gamma(n+1) - 2\beta. \tag{4.14}$$

To determine γ and β uniquely we need another relation. In §4.3 we demonstrated that the PME conserves mass giving

$$\int_0^{R(t)} r^{\delta-1}u(r,t)\ \mathrm{d}r = k, \tag{4.15}$$

where k is a constant, in the radially symmetric case. Transforming (4.15) to the variables $\hat{u}$, $\hat{t}$ and $\hat{r}$, by (4.11),

$$\int_0^{R(\hat{t})} \lambda^{\gamma+\beta\delta}\hat{r}^{\delta-1}\hat{u}(\hat{r},\hat{t})\ \mathrm{d}\hat{r} = k.$$

Equating the λ terms on either side,

$$\lambda^{\gamma+\beta\delta} = \lambda^0.$$

Consequently, if

$$\gamma + \beta\delta = 0. \tag{4.16}$$

then (4.15) is invariant. Solving (4.14) and (4.16) simultaneously we find that

$$\beta = \frac{1}{n\delta+2} \qquad \text{and} \qquad \gamma = -\frac{\delta}{n\delta+2}, \tag{4.17}$$

and the scale invariant transformation (4.11) becomes

$$t = \lambda\hat{t}, \qquad r = \lambda^{\frac{1}{n\delta+2}}\hat{r}, \qquad u = \lambda^{-\frac{\delta}{n\delta+2}}\hat{u}. \tag{4.18}$$

To summarise, the variables u, r and t can be rescaled as in (4.18) whilst still satisfying the PME (4.7) independently of the scaling parameter λ.

We now define scale-invariant similarity variables. From (4.11) we have

$$\lambda = \frac{t}{\hat{t}} = \frac{r^{\frac{1}{\beta}}}{\hat{r}^{\frac{1}{\beta}}} = \frac{u^{\frac{1}{\gamma}}}{\hat{u}^{\frac{1}{\gamma}}}.$$

Bearing this rescaling in mind, we introduce the two new variables,

$$\begin{aligned} \zeta &= \frac{u}{t^\gamma} = \frac{\hat{u}}{\hat{t}^\gamma}, && (4.19) \\ \xi &= \frac{r}{t^\beta} = \frac{\hat{r}}{\hat{t}^\beta}, && (4.20) \end{aligned}$$

which are independent of λ and hence scale invariant under the transformation (4.11). We use (4.19) and (4.20) to find a self-similar solution for the PME (4.7) in the next section.

4.4.2 Self-similarity

Self-similarity occurs when the *solution* of the problem (as opposed to the PDE) is invariant under the scaling transformation, and is a property of the PME which our moving mesh method preserves [9]. We obtain a self-similar solution of the PME (4.7) by assuming that there is a functional relationship $\zeta = \zeta(\xi)$ between the similarity variables (4.19) and (4.20), based on our rescaling (4.11). The self-similar solution is then dependent only on the solution to an ordinary differential equation (ODE) that is obtained by transforming the PME (4.7) into the variables ζ and ξ with β and γ defined by (4.17).

First, we transform the left-hand side of the PME,

$$\begin{aligned} \frac{\partial u}{\partial t} &= \frac{\partial}{\partial t}(\zeta(\xi)t^\gamma), \\ &= t^\gamma \frac{\partial \zeta}{\partial t} + \zeta(\xi)\gamma t^{\gamma-1}, \\ &= t^\gamma \frac{\partial \xi}{\partial t}\frac{\mathrm{d}\zeta}{\mathrm{d}\xi} + \zeta(\xi)\gamma t^{\gamma-1}. \end{aligned}$$

Substituting $\frac{\partial \xi}{\partial t} = \frac{\partial}{\partial t}(rt^{-\beta}) = -\beta r t^{-(\beta+1)}$, from (4.20), in the first term on the right-hand side gives

$$\frac{\partial u}{\partial t} = -\beta r t^{\gamma-(\beta+1)}\frac{\mathrm{d}\zeta}{\mathrm{d}\xi} + \zeta(\xi)\gamma t^{\gamma-1}.$$

A further substitution of $rt^{-\beta} = \xi$, from (4.20), in the first term on the right-hand side gives

$$\frac{\partial u}{\partial t} = -\beta t^{\gamma-1}\xi\frac{\mathrm{d}\zeta}{\mathrm{d}\xi} + \zeta(\xi)\gamma t^{\gamma-1}. \tag{4.21}$$

We have expressed the left-hand side of (4.7) in terms of ζ and ξ, and we wish to transform the right-hand side in the same way. Thus

$$\frac{1}{r^{\delta-1}}\frac{\partial}{\partial r}\left(r^{\delta-1}u(r,t)^n\frac{\partial u}{\partial r}\right) = t^{\beta(1-\delta)}\frac{1}{\xi^{\delta-1}}\frac{\partial \xi}{\partial r}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\xi^{\delta-1}t^{\beta(\delta-1)}\zeta(\xi)^n t^{\gamma n}\frac{\partial \xi}{\partial r}\frac{\partial u}{\partial \zeta}\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}\right).$$

Substituting $\frac{\partial \xi}{\partial r} = t^{-\beta}$ from (4.20), and $\frac{\partial u}{\partial \zeta} = t^{\gamma}$ from (4.19) into the right-hand side, gives

$$\begin{aligned}\frac{1}{r^{\delta-1}}\frac{\partial}{\partial r}\left(r^{\delta-1}u(r,t)^n\frac{\partial u}{\partial r}\right) &= \frac{1}{\xi^{\delta-1}}t^{-\beta\delta}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\xi^{\delta-1}t^{\beta(\delta-2)+\gamma(n+1)}\zeta(\xi)^n\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}\right), \\ &= t^{\gamma(n+1)-2\beta}\frac{1}{\xi^{\delta-1}}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\xi^{\delta-1}\zeta(\xi)^n\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}\right). \qquad (4.22)\end{aligned}$$

Putting together (4.21) and (4.22) gives the PME (4.7) in terms of ζ and ξ,

$$-\beta t^{-1}\xi\frac{\mathrm{d}\zeta}{\mathrm{d}\xi} + \zeta(\xi)\gamma t^{-1} = t^{n\gamma-2\beta}\frac{1}{\xi^{\delta-1}}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\xi^{\delta-1}\zeta(\xi)^n\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}\right).$$

Note that t disappears from the equation since $n\gamma - 2\beta + 1 = 0$ (from (4.14)). Substituting for β and γ from (4.17), gives the ODE

$$-\frac{1}{n\delta+2}\xi\frac{\mathrm{d}\zeta}{\mathrm{d}\xi} - \zeta(\xi)\left(\frac{\delta}{n\delta+2}\right) = \frac{1}{\xi^{\delta-1}}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\xi^{\delta-1}\zeta(\xi)^n\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}\right).$$

By moving all the terms to one side we have,

$$\frac{1}{\xi^{\delta-1}}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\xi^{\delta-1}\zeta(\xi)^n\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}\right) + \frac{\xi}{n\delta+2}\frac{\mathrm{d}\zeta}{\mathrm{d}\xi} + \frac{\zeta(\xi)\delta}{n\delta+2} = 0. \tag{4.23}$$

From the zero Dirichlet boundary conditions imposed on the PME we have corresponding zero Dirichlet boundary conditions on ζ in (4.23). The solution of the two point boundary problem (4.23), along with the previous definitions $u = \zeta t^{\gamma}$ and $r = \xi t^{\beta}$ provides the self-similar solution.

Using an integrating factor $\left\{\exp\left(\int \frac{\delta}{\xi}\,\mathrm{d}\xi\right) = \xi^{\delta}\right\}$ enables us to group the last two terms of (4.23), giving

$$\frac{1}{\xi^{\delta-1}}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\xi^{\delta-1}\zeta(\xi)^n\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}\right) + \frac{1}{(n\delta+2)\xi^{\delta-1}}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\xi^{\delta}\zeta(\xi)\right) = 0.$$

We multiply through by $\xi^{\delta-1}$ and integrate to get

$$\xi^{\delta-1}\zeta(\xi)^n\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}+\frac{\xi^\delta\zeta(\xi)}{(n\delta+2)}=C,$$

where C is an integration constant. Taking $C=0$, which it is at the boundary where $\zeta=0$,

$$\xi^{-1}\zeta(\xi)^{n-1}\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}+\frac{1}{n\delta+2}=0.$$

Separating the variables,

$$\int\zeta(\xi)^{n-1}\,\mathrm{d}\zeta=-\frac{1}{n\delta+2}\int\xi\,\mathrm{d}\xi,$$

which gives

$$\frac{\zeta(\xi)^n}{n}=-\frac{1}{n\delta+2}\left(\frac{\xi^2}{2}-E\right),$$

where E is a constant of integration. Bearing in mind that $u=\zeta t^\gamma$ (from (4.19)), we make ζ the subject and, setting $A_n=\frac{nE}{n\delta+2}$,

$$\zeta(\xi)=\left(A_n-\frac{n}{2(n\delta+2)}\xi^2\right)^{\frac{1}{n}}.$$

Hence, there exists the following self-similar solution within a moving compact interval for which $u=0$ at the boundary,

$$\zeta(\xi)=\begin{cases}\left(A_n-\frac{n}{2(n\delta+2)}\xi^2\right)^{\frac{1}{n}} & \frac{n}{2(n\delta+2)}\xi^2\leq A_n,\\ 0 & \frac{n}{2(n\delta+2)}\xi^2>A_n.\end{cases}$$

Mapping this back to the original variables u, r and t using the definitions (4.19)–(4.20), with β and γ given by (4.17), we achieve

$$u(r,t)=\frac{1}{t^{\frac{\delta}{n\delta+2}}}\left(A_n-\frac{nr^2}{2(n\delta+2)t^{\frac{2}{n\delta+2}}}\right)_+^{\frac{1}{n}},\tag{4.24}$$

where the notation $(\cdot)_+^{\frac{1}{n}}$ indicates that we take the positive solution of $(\cdot)^{\frac{1}{n}}$. Equation (4.24) is an exact self-similar source solution for the general PME (4.7) with zero boundary conditions (due originally to [12] and [77]). A diagrammatic representation of $u(r,t)$ in the one-dimesional Cartesian case for $n>1$ (where the gradient at the boundaries is infinite) is shown in Figure 4.1.

When presenting our numerical results we will mainly use this self-similar solution as initial conditions. We also discuss the effect of not using a self-similar solution as the initial conditions, where we find that the PME boundaries do not move until the solution at the boundary resembles a self-similar solution. This behaviour is nicely captured by our moving mesh scheme.

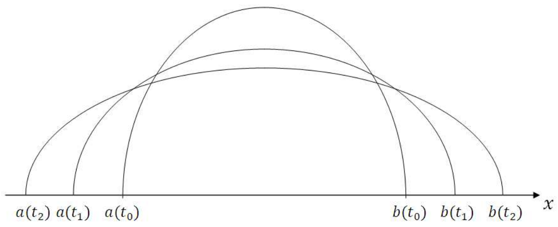


Fig. 4.1: Diagrammatic representation of the PME.

4.4.3 A specific set of parameters

For the purpose of numerical comparisons we consider (4.24) with $A_n = 1$. At the boundaries $u = 0$ therefore, the boundary $R(t)$ is given by

$$R(t) = \sqrt{\frac{2(n\delta + 2)t^{\frac{2}{n\delta+2}}}{n}}, \tag{4.25}$$

where $[0, R(t)]$ is the support of u. Substituting this definition back into (4.24) we have

$$u(r,t) = \frac{1}{t^{\frac{\delta}{n\delta+2}}} \left(1 - \frac{r^2}{R(t)^2}\right)_+^{\frac{1}{n}}. \tag{4.26}$$

At time $t = 1$ we have

$$u(r,1) = \left(1 - \frac{r^2}{R(1)^2}\right)_+^{\frac{1}{n}}, \tag{4.27}$$

where

$$R(1) = \sqrt{\frac{2(n\delta + 2)}{n}}. \tag{4.28}$$

We use (4.27) and (4.28) as initial conditions for much of our numerical comparison work.

We have derived a self-similar solution for the PME. This provides us with an initial solution for our numerical work, as well as an exact solution to compare with our numerical results. In the next section we apply our moving mesh method to the Cartesian one-dimensional PME.

4.5 Moving meshes

The integral of a solution of the PME (the mass) is conserved in time, so we use the moving mesh method described in §3.1, with the same notation, i.e. $\tilde{x}_j(t^m) \approx x_j^m$ denotes the jth node of the mesh with $N+1$ nodes, at time $m\Delta t$, $m = 0, 1 \ldots$, and $u_j^m \approx \tilde{u}_j(t^m)$ and $v_j^m \approx \tilde{v}_j(t^m)$ denote the solution and mesh velocity at these nodes.

We model the PDE (3.1) with

$$\mathcal{L}u \equiv \frac{\partial}{\partial x}\left(u(x,t)^n \frac{\partial u}{\partial x}\right), \tag{4.29}$$

from (4.6). The moving mesh method in §3.1 can be applied to any geometrically non-symmetric problem. However, for convenience we assume that the one-dimensional solution $u(x,t)$ is symmetrical about its centre of mass (see §4.3). Then by symmetry we need only model half the region $x(t) \in [0, b(t)]$. For the symmetrical problem, the boundary conditions for u are

$$\frac{\partial u}{\partial x} = 0 \qquad \text{at} \quad x = x_0 = 0, \tag{4.30}$$

$$u = 0 \qquad \text{at} \quad x = b(t), \tag{4.31}$$

where x_0 is fixed but $b(t)$ moves with time.

Using this method we show that, given a mesh $\tilde{x}_j(t^m)$, with corresponding solution $\tilde{u}_j(t^m)$, we can calculate the updated mesh $\tilde{x}_j(t^{m+1})$ and solution $\tilde{u}_j(t^{m+1})$ by computing a mesh velocity $\tilde{v}_j(t^m)$.

4.5.1 Determining the mesh velocity

The mesh velocity is given by substituting (4.29) into (3.8) where $a(t) = 0$,

$$\begin{aligned} \tilde{v}_j(t) &= -\frac{1}{\tilde{u}_j(t)} \int_0^{\tilde{x}_j(t)} \frac{\partial}{\partial x}\left(u(x,t)^n \frac{\partial u}{\partial x}\right) \mathrm{d}x, \\ &= -\frac{1}{\tilde{u}_j(t)} \left[u(x,t)^n \frac{\partial u}{\partial x}\right]_0^{\tilde{x}_j(t)}, \end{aligned}$$

for interior points $j = 1, \ldots, N-1$. Since $\frac{\partial u}{\partial x} = 0$ at $x = 0$, the interior points move in time such that

$$\tilde{v}_j(t) = -\tilde{u}_j(t)^{n-1} \frac{\partial u}{\partial x}\bigg|_{\tilde{x}_j(t)}, \tag{4.32}$$

which can also be written as

$$\tilde{v}_j(t) = -\frac{1}{n} \frac{\partial (u^n)}{\partial x}\bigg|_{\tilde{x}_j(t)}. \tag{4.33}$$

Remark 4.5.1 *We refer to §4.2 and observe that the mesh velocity (4.33) resembles the velocity* $\mathbf{V}$ *obtained from substituting the equation of state (4.4) into Darcy's Law (4.3),*

$$\mathbf{V} = -\frac{\kappa p_0}{\mu} \nabla \rho^{\gamma},$$

where $\rho = u$ *and* $\gamma = n$.

We use a discretised form of (4.33), at time $t = t^m$

$$v_j^m = -\frac{1}{n} \frac{(u^n)_{j+1}^m - (u^n)_{j-1}^m}{(x_{j+1}^m - x_{j-1}^m)}, \qquad j = 1, 2, ..., N-1, \tag{4.34}$$

which is a second order discretisation on a uniform mesh, but only a first order discretisation on a non-uniform mesh. By definition, at the inner boundary $v_0^m = 0$. The outer boundary velocity v_N^m is extrapolated by a polynomial approximation using $(v_{N-3}^m, v_{N-2}^m, v_{N-1}^m)$.

We note that when approximating $(u^n)_x\big|_{\tilde{x}_j(t)} = \frac{\partial(u^n)}{\partial x}\bigg|_{\tilde{x}_j(t)}$ in (4.34), we do not use the known value $(u^n)_j^m$. We can improve upon (4.34) by developing a second order approximation for $(u^n)_x\big|_{\tilde{x}_j(t)}$ which uses the three values $(u^n)_{j-1}^m$, $(u^n)_j^m$ and $(u^n)_{j+1}^m$. This is a general form of the approximation for u_x given in [76].

We define $\Delta x_{j+}^m = x_{j+1}^m - x_j^m$ and $= \Delta x_{j-}^m = x_j^m - x_{j-1}^m$ and write $u(x,t)^n$ at the nodes $(j-1)$ and $(j+1)$ at time t^m as

$$\begin{aligned} (u^n)_{j+1}^m &\approx u(\tilde{x}_j + \Delta x_{j+}^m, t)^n, \\ (u^n)_{j-1}^m &\approx u(\tilde{x}_j - \Delta x_{j-}^m, t)^n. \end{aligned}$$

Expanding both of these using a Taylor series about $\tilde{x}_j(t)$ gives

$$(u^n)_{j+1}^m \approx \tilde{u}_j(t)^n + \Delta x_{j+}^m \left.\frac{\partial(u^n)}{\partial x}\right|_{\tilde{x}_j(t)} + \frac{1}{2}(\Delta x_{j+}^m)^2 \left.\frac{\partial^2(u^n)}{\partial x^2}\right|_{\tilde{x}_j(t)} + \mathcal{O}(\Delta x_{j+}^m)^3, \quad (4.35)$$

$$(u^n)_{j-1}^m \approx \tilde{u}_j(t)^n - \Delta x_{j-}^m \left.\frac{\partial(u^n)}{\partial x}\right|_{\tilde{x}_j(t)} + \frac{1}{2}(\Delta x_{j-}^m)^2 \left.\frac{\partial^2(u^n)}{\partial x^2}\right|_{\tilde{x}_j(t)} + \mathcal{O}(\Delta x_{j-}^m)^3. \quad (4.36)$$

We subtract (4.36) multiplied by $(\Delta x_{j+}^m)^2$ from (4.35) multiplied by $(\Delta x_{j-}^m)^2$ to give a higher order approximation to $(u^n)_x\big|_{\tilde{x}_j(t)}$ at time t^m as,

$$\begin{aligned}(u^n)_x\big|_{\tilde{x}_j(t)} = \left.\frac{\partial(u^n)}{\partial x}\right|_{\tilde{x}_j(t)} &\approx \frac{(\Delta x_{j-}^m)^2\left[(u^n)_{j+1}^m - (u^n)_j^m\right] + (\Delta x_{j+}^m)^2\left[(u^n)_j^m - (u^n)_{j-1}^m\right]}{\Delta x_{j-}^m\,\Delta x_{j+}^m\left[\Delta x_{j+}^m + \Delta x_{j-}^m\right]}, \\ &= \frac{\frac{1}{\Delta x_{j+}^m}\left(\frac{\Delta(u^n)_{j+}^m}{\Delta x_{j+}^m}\right) + \frac{1}{\Delta x_{j-}^m}\left(\frac{\Delta(u^n)_{j-}^m}{\Delta x_{j-}^m}\right)}{\frac{1}{\Delta x_{j+}^m} + \frac{1}{\Delta x_{j-}^m}}, \end{aligned} \quad (4.37)$$

for $j = 1, \ldots, N-1$, where $\Delta(u^n)_{j+}^m = (u^n)_{j+1}^m - (u^n)_j^m$ and $\Delta(u^n)_{j-}^m = (u^n)_j^m - (u^n)_{j-1}^m$ (corresponding to our earlier definitions for $\Delta x_{j\pm}^m$). The second order approximation (4.37) is an inversely weighted sum, or interpolation, of slopes and is frequently used in numerical work throughout this thesis.

Substituting (4.37) in (4.33) gives the second-order approximation to the mesh velocity,

$$v_j^m = -\frac{1}{n}\frac{\frac{1}{\Delta x_{j+}^m}\left(\frac{\Delta(u^n)_{j+}^m}{\Delta x_{j+}^m}\right) + \frac{1}{\Delta x_{j-}^m}\left(\frac{\Delta(u^n)_{j-}^m}{\Delta x_{j-}^m}\right)}{\frac{1}{\Delta x_{j+}^m} + \frac{1}{\Delta x_{j-}^m}}, \quad (4.38)$$

for $j = 1, \ldots, N-1$ where the time-level m notation has been re-instated. As with (4.34), the velocity at the inner boundary is given by $v_0^m = 0$, and at the outer boundary the velocity v_N^m is extrapolated by a polynomial approximation using $(v_{N-3}^m, v_{N-2}^m, v_{N-1}^m)$.

The new mesh x_j^{m+1} is obtained from v_j^m by a time-stepping scheme.

4.5.2 Recovering the solution

Once the updated mesh x_j^{m+1} has been determined, the updated solution u_j^{m+1}, $j = 1, \ldots, N-1$, is given by either (3.10) or (3.16), the latter being exact for linear u_j on a non-uniform mesh. The solution at the inner boundary u_0^{m+1} is calculated using

$$u_0^{m+1} = \frac{x_1^0}{x_1^{m+1}}u_0^0,$$

from (3.10) and the boundary condition (4.30). At the outer boundary, $u_{N+1}^{m+1} = 0$ from (4.31).

4.5.3 The full algorithm

Given a mesh x_j^m, solution u_j^m, $j = 0, \ldots, N$, $m \geq 0$:

- Compute the mesh velocity v_j^m from (4.34) or (4.38);
- Using a time-stepping scheme, compute the updated mesh x_j^{m+1} by a time-stepping scheme;
- Compute the updated solution u_j^{m+1} from (3.10) or (3.16).

4.5.4 Waiting times

The velocity of the boundary is given by (4.32). The boundary behaviour of the PME has been investigated in [62, 90]. We now present some of their findings.

At the boundary $u = 0$, which infers that the boundary velocity is zero unless u_x is infinite for $n > 1$. We examine the effect of the initial conditions on the boundary velocity by considering initial conditions at $t = 1$, of the form given in [18, 62],

$$u(x,1) \;=\; \left[1 - \left(\frac{\tilde{x}(1)}{b(1)}\right)^2\right]^\alpha = \left(1 - \frac{\tilde{x}(1)}{b(1)}\right)^\alpha \left(1 + \frac{\tilde{x}(1)}{b(1)}\right)^\alpha, \tag{4.39}$$

where $\alpha = \frac{1}{n}$ (the self-similar initial condition). We use (4.39) to determine u_x in (4.32), to give

$$\frac{\mathrm{d}\tilde{x}}{\mathrm{d}t} = \frac{2\alpha x}{b(1)^2}\left(1 - \frac{\tilde{x}(1)}{b(1)}\right)^{\alpha n - 1}\left(1 + \frac{\tilde{x}(1)}{b(1)}\right)^{\alpha n - 1}. \tag{4.40}$$

We consider equation (4.40) as $\tilde{x}(t) \to b(t)$, and distinguish three cases:

1. If $\alpha n > 1$, then $\frac{\mathrm{d}b}{\mathrm{d}t} = 0$, indicating that the boundary does not move. The period of time that boundary does not move is referred to as the 'waiting time';
2. If $\alpha n < 1$, then $\frac{\mathrm{d}b}{\mathrm{d}t} \to \infty_+$ indicating that $b(1)$ initially moves with infinite velocity, but the velocity quickly becomes finite;
3. If $\alpha n = 1$ (i.e. the self-similar case), then the boundary has velocity

$$\frac{\mathrm{d}b}{\mathrm{d}t} = \frac{2\alpha}{b(1)}.$$

The boundary behaviour for these three cases is shown in Figure 4.2.

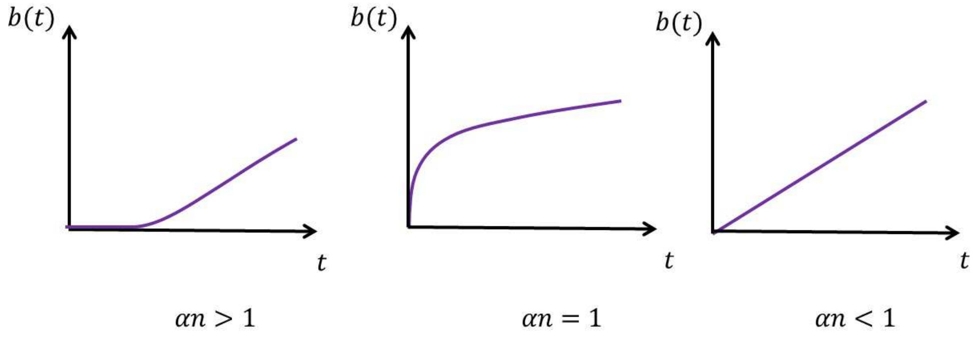


Fig. 4.2: The three different types of boundary behaviour for the PME.

4.5.5 Time-stepping schemes

Explicit schemes

The simplest method to time-step the mesh is the first order explicit Euler time-stepping scheme,

$$\frac{x_j^{m+1} - x_j^m}{\Delta t} = v_j^m, \tag{4.41}$$

for $j = 1, \ldots, N-1$. We substitute for v_j^m from (4.34) or (4.38). At the inner boundary $v_0^m = 0$, and at the outer boundary we use a polynomial approximation to determine v_N^m and substitute it into (4.41) for $j = N$. Alternatively, we could use the one-sided approximation of (4.34) to give

$$\frac{x_N^{m+1} - x_N^m}{\Delta t} = -\frac{1}{n}\frac{(u^n)_N^m - (u^n)_{N-1}^m}{x_N^m - x_{N-1}^m}, \tag{4.42}$$

but this is of lower order accuracy.

The explicit Euler time-stepping scheme requires small Δt so that the x_j^m do not tangle, i.e. (3.5) holds. We also implemented the adaptive Runge-Kutta methods in Matlab with similar results. In addition, we used the solver ODE15s, which is designed to solve a stiff system. Implementing this solver produced results similar to results from ODE23 and ODE45 (which are not designed for stiff systems), inferring that the method does not lead to stiff systems in this case.

A semi-implicit scheme

To determine a semi-implicit time-stepping scheme for solving the PME we consider the general semi-implicit time-stepping scheme (3.39) (for the internal nodes, $j = 1, 2, ..., N-1$) with

$$\phi^m_{j_+} = -\frac{1}{n}(u^n)^m_{j+\frac{1}{2}}, \qquad \phi^m_{j_-} = -\frac{1}{n}(u^n)^m_{j-\frac{1}{2}},$$

from (4.33). To ensure that the mesh does not tangle using (3.40), we determine the $\Delta x^{m+1}_{j_\pm}$ terms in (3.39) such that

$$\frac{x^{m+1}_j - x^m_j}{\Delta t} = -\frac{1}{n\Delta x^m_j}\left((u^n)^m_{j+\frac{1}{2}}\frac{\Delta x^{m+1}_{j_-}}{\Delta x^m_{j_-}} - (u^n)^m_{j-\frac{1}{2}}\frac{\Delta x^{m+1}_{j_+}}{\Delta x^m_{j_+}}\right), \tag{4.43}$$

where $\Delta x^m_j = x^m_{j+\frac{1}{2}} - x^m_{j-\frac{1}{2}}$, $\Delta x^m_{j_-} = x^m_j - x^m_{j-1}$ and $\Delta x^m_{j_+} = x^m_{j+1} - x^m_j$. Before calculating the internal nodes semi-implicitly by (4.43), the boundary node x^{m+1}_N is calculated by the explicit scheme (4.42), enabling $\Delta x^{m+1}_{N-1_+} = x^{m+1}_N - x^{m+1}_{N-1}$ to be determined.

Rearranging (4.43) and expanding the $\Delta x^{m+1}_{j_\pm}$ terms,

$$\frac{x^{m+1}_j - x^m_j}{\Delta t} = -\frac{(u^n)^m_{j+\frac{1}{2}}\Delta x^m_{j_+}(x^{m+1}_j - x^{m+1}_{j-1}) - (u^n)^m_{j-\frac{1}{2}}\Delta x^m_{j_-}(x^{m+1}_{j+1} - x^{m+1}_j)}{n\Delta x^m_j \Delta x^m_{j_+}\Delta x^m_{j_-}}. \tag{4.44}$$

Our moving mesh method moves the nodes such that partial masses of the solution are conserved, see equation (3.6) in §3.1. Bearing this in mind we define approximations to the mass in the subintervals which remain unchanged in time, as

$$\begin{aligned} C_{j_+} &= (u^n)^0_{j+\frac{1}{2}}\Delta x^0_{j_+} = (u^n)^m_{j+\frac{1}{2}}\Delta x^m_{j_+}, \\ C_{j_-} &= (u^n)^0_{j-\frac{1}{2}}\Delta x^0_{j_-} = (u^n)^m_{j-\frac{1}{2}}\Delta x^m_{j_-}, \end{aligned}$$

thus simplifying equation (4.44) to

$$\frac{x^{m+1}_j - x^m_j}{\Delta t} = -\frac{C_{j_+}(x^{m+1}_j - x^{m+1}_{j-1}) - C_{j_-}(x^{m+1}_{j+1} - x^{m+1}_j)}{n\Delta x^m_j \Delta x^m_{j_+}\Delta x^m_{j_-}}. \tag{4.45}$$

To determine the new mesh by the semi-implicit scheme we solve the system

$$A\mathbf{x}^{m+1} = \mathbf{x}^m, \tag{4.46}$$

where $\mathbf{x}^{m+1} = (x_1^{m+1}, \cdots x_{N-1}^{m+1})^T$, $\mathbf{x}^m = (x_1^m, \cdots x_{N-1}^m)^T$, and A is a tridiagonal matrix with lower, main and upper diagonals Al_j, Ad_j and Au_j given by

$$\begin{aligned} Al_j &= -\frac{C_{j+}\Delta t}{n\Delta x_j^m \Delta x_{j+}^m \Delta x_{j-}^m}, \\ Au_j &= -\frac{C_{j-}\Delta t}{n\Delta x_j^m \Delta x_{j+}^m \Delta x_{j-}^m}, \\ Ad_j &= 1 - Al_j - Au_j. \end{aligned}$$

Remark 4.5.2 *The implicitness of the semi-implicit approach, together with the explicit end point calculation, can be improved by using it in a predictor-corrector mode: solving the matrix system (4.46) repeatedly within one time-level, whilst updating the explicit end point value x_N^{m+1}. The corrections affect entries of the matrix A and the vector $\mathbf{x}^{m+1}$, but the right-hand side vector $\mathbf{x}^m$ remain unchanged. We use the notation $(\cdot)^p$ to denote the iterations. After one iteration ($p = 1$) the x entries of A are at the new time-level $(m+1)$, giving $(x_j^{m+1})^1$ terms, and the entries of $\mathbf{x}^{m+1}$ are $(x_j^{m+1})^2$. This additional iteration process at each time-level modifies (4.45) to become*

$$\frac{(x_j^{m+1})^{p+1} - x_j^m}{\Delta t} = -\frac{C_{j+}(x_j^{m+1} - x_{j-1}^{m+1})^{p+1} - C_{j-}(x_{j+1}^{m+1} - x_j^{m+1})^{p+1}}{n(\Delta x_j^{m+1})^p (\Delta x_{j+}^{m+1})^p (\Delta x_{j-}^{m+1})^p}, \tag{4.47}$$

where $p = 0, 1, 2, \ldots$. Note that the x_j^m term on the left-hand side remains unchanged during this additional iteration process. If (4.47) is convergent it leads to the fully implicit scheme

$$\frac{x_j^{m+1} - x_j^m}{\Delta t} = -\frac{1}{n\Delta x_j^{m+1}}\left(\frac{C_{j+}}{\Delta x_{j+}^{m+1}} - \frac{C_{j-}}{\Delta x_{j-}^{m+1}}\right).$$

In view of the time we did not implement this scheme here. A different fully implicit scheme has however been successfully programmed up by Scherer-Abreu [87].

We have given details of applying the moving mesh method of §3.1 to the one-dimensional PME. We now demonstrate that the same method can be applied to the two-dimensional radially symmetric PME.

4.6 The radially symmetric case

We use the moving mesh method described in §3.1 to approximate the radially symmetric PME (4.7) in δ dimensions,

$$\frac{\partial u}{\partial t} = \frac{1}{r^{\delta-1}}\frac{\partial}{\partial r}\left(r^{\delta-1}u(r,t)^n\frac{\partial u}{\partial r}\right). \tag{4.48}$$

with boundary conditions

$$\frac{\partial u}{\partial r} = 0 \qquad \text{at} \quad r = r_0 = 0, \tag{4.49}$$

$$u = 0 \qquad \text{at} \quad r = R(t), \tag{4.50}$$

where $R(t)$ moves with time.

We slightly vary the notation from §3.1 such that the space variable x is replaced by r and v replaced by s. We use $\tilde{r}_j(t^m) \approx r_j^m$, $j = 0, 1, \ldots, N+1$, to denote the jth node along the radius of the mesh, at time $m\Delta t$, $m = 0, 1 \ldots$. The corresponding solution notation is $u(\tilde{r}_j, t^m) = \tilde{u}_j(t^m) \approx u_j^m$, and the velocity of each node is

$$s(\tilde{r}_j, t^m) = \tilde{s}_j(t) = \frac{\mathrm{d}\tilde{r}_j}{\mathrm{d}t}.$$

We use a similar procedure to the one given carried out in §3.1, to show that, given a mesh along a radius $\tilde{r}_j(t^m)$, with corresponding solution $\tilde{u}_j(t^m)$, the updated mesh along a radius $\tilde{r}_j(t^{m+1})$ and solution $\tilde{u}_j(t^{m+1})$ can be found by computing a mesh velocity $\tilde{s}_j(t^m)$.

4.6.1 Determining the mesh velocity

As with the one-dimensional case we seek a mesh velocity by differentiating the total mass with respect to time, using the Leibnitz integral rule,

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_0^{r_j(t)} r^{\delta-1}u(r,t)\,\mathrm{d}r = \int_0^{\tilde{r}_j(t)} r^{\delta-1}\frac{\partial u}{\partial t}\,\mathrm{d}r + \left[r^{\delta-1}u(r,t)\frac{\mathrm{d}r}{\mathrm{d}t}\right]_0^{\tilde{r}_j(t)} = 0.$$

Substituting for $\frac{\partial u}{\partial t}$ from (4.48), evaluating the integral, and cancelling terms due to the boundary conditions (4.49) gives

$$\tilde{r}_j(t)^{\delta-1}\tilde{u}_j(t)^n\left.\frac{\partial u}{\partial r}\right|_{\tilde{r}_j(t)} + \tilde{r}_j(t)^{\delta-1}\tilde{u}_j(t)\tilde{s}_j(t) = 0.$$

Hence, the mesh velocity is given by

$$\begin{aligned}\tilde{s}_j(t) &= -\tilde{u}_j(t)^{n-1}\frac{\partial u}{\partial r}\bigg|_{\tilde{r}_j(t)}, \\ &= -\frac{1}{n}\frac{\partial(u^n)}{\partial r}\bigg|_{\tilde{r}_j(t)}. \end{aligned} \tag{4.51}$$

We note that the mesh velocity is independent of the number of dimensions δ.

To approximate (4.51) we use either the first order discrete form,

$$s_j^m = -(u^{n-1})_j^m \frac{u_{j+1}^m - u_{j-1}^m}{r_{j+1}^m - r_{j-1}^m}, \tag{4.52}$$

for $j = 1, \dots, N-1$, or the second order discrete form from (4.37),

$$s_j^m = -(u^{n-1})_j^m \frac{\frac{1}{\Delta r_{j_+}^m}\left(\frac{\Delta(u^n)_{j_+}^m}{\Delta r_{j_+}^m}\right) + \frac{1}{\Delta r_{j_-}^m}\left(\frac{\Delta(u^n)_{j_-}^m}{\Delta r_{j_-}^m}\right)}{\frac{1}{\Delta r_{j_+}^m} + \frac{1}{\Delta r_{j_-}^m}}, \tag{4.53}$$

for $j = 1, \dots, N-1$, where $\Delta(\cdot)_{j_+}^m = (\cdot)_{j+1}^m - (\cdot)_j^m$ and $\Delta(\cdot)_{j_-}^m = (\cdot)_j^m - (\cdot)_{j-1}^m$. At the inner boundary $s_0^m = 0$ by definition, and the outer boundary velocity s_N^m is extrapolated by a polynomial approximation using $(s_{N-3}^m, s_{N-2}^m, s_{N-1}^m)$.

As with the one-dimensional case, the mesh velocity is used with a time-stepping scheme to update the mesh at each time-level.

4.6.2 Recovering the solution

When recovering the solution $\tilde{u}_j(t)$ on the new mesh we use the conservation of mass principle in the form

$$\int_{r_{j-1}^{m+1}}^{r_{j+1}^{m+1}} r^{\delta-1}u(r, t^{m+1})\,\mathrm{d}r = \int_{r_{j-1}^0}^{r_{j+1}^0} r^{\delta-1}u(r, t^0)\,\mathrm{d}r, \quad j = 1, \dots, N-1, \tag{4.54}$$

where t^0 is the initial time, which is a generalised form of (3.9). The right-hand side of (4.54) is known from the initial data. We approximate the integrals of (4.54) using a quadrature. Evaluating the first integral at $t = t^{m+1}$ allows us to recover u_j^{m+1}, $j = 0, \dots, N$. As in §3.1, we examine two different quadrature rules, a mid-point approximation and an interpolation approximation on a non-uniform mesh.

A simple mid-point approximation of (4.54) gives

$$\left[r_{j+1}^{m+1} - r_{j-1}^{m+1}\right](r^{\delta-1})_j^{m+1}u_j^{m+1} = \left[r_{j+1}^0 - r_{j-1}^0\right](r^{\delta-1})_j^0 u_j^0,$$

and hence means that the updated solution is recovered on the new mesh by

$$u_j^{m+1} = \frac{(r^{\delta-1})_j^0}{(r^{\delta-1})_j^{m+1}} \left(\frac{r_{j+1}^0 - r_{j-1}^0}{r_{j+1}^{m+1} - r_{j-1}^{m+1}} u_j^0 \right) \tag{4.55}$$

for $j = 1, \cdots, N-1$.

However, as the mesh is not necessarily uniform, it is more accurate to use a quadrature rule for a non-uniform mesh. We note that the term in the brackets on the right-hand side of (4.55) is the same as the right-hand side of (3.10), where $x = r$. Subsequently, we can replace this term in (4.55) with the right-hand side of (3.16) (substituting $x = r$), which simplifies to (4.55) for a uniform mesh. Hence, a better approximation to update the solution is

$$u_j^{m+1} = \frac{(r^{\delta-1})_j^0}{(r^{\delta-1})_j^{m+1}} \left(\frac{\frac{c_{j_-}/\Delta r_{j_-}^{m+1}}{\Delta r_{j_-}^{m+1}} + \frac{c_{j_+}/\Delta r_{j_+}^{m+1}}{\Delta r_{j_+}^{m+1}}}{\frac{1}{\Delta r_{j_-}^{m+1}} + \frac{1}{\Delta r_{j_+}^{m+1}}} \right), \tag{4.56}$$

for $j = 1, \ldots, N-1$, where

$$\begin{aligned} c_{j_-} &= \int_{\tilde{r}_{j-1}(t^0)}^{\tilde{r}_j(t^0)} u(r,t^0) r^{\delta-1}\,\mathrm{d}r = \int_{\tilde{r}_{j-1}(t)}^{\tilde{r}_j(t)} u(r,t) r^{\delta-1}\,\mathrm{d}r, \\ c_{j_+} &= \int_{\tilde{r}_j(t^0)}^{\tilde{r}_{j+1}(t^0)} u(r,t^0) r^{\delta-1}\,\mathrm{d}r = \int_{\tilde{r}_j(t)}^{\tilde{r}_{j+1}(t)} u(r,t) r^{\delta-1}\,\mathrm{d}r, \end{aligned}$$

are calculated from the initial data.

For $j = 0$, (4.55) and (4.56) break down since $r_0^m = 0$. However, at the inner boundary u is locally quadratic since $\frac{\partial u}{\partial r} = 0$, hence we use the approximation

$$u(r,t) \approx \tilde{u}_0(t) + r^2 k,$$

for $r \approx 0$, where k is a (small) constant denoting the curvature. Substituting this approximation into (4.54) for $j = 0$ gives

$$\int_{\tilde{r}_{-1}(t)}^{\tilde{r}_1(t)} r^{\delta-1} \left(\tilde{u}_0(t) + r^2 k \right)\,\mathrm{d}r \quad = \quad \int_{\tilde{r}_{-1}(0)}^{\tilde{r}_1(0)} r^{\delta-1} \left(\tilde{u}_0(0) + r^2 k \right)\,\mathrm{d}r,$$

where $\tilde{r}_{-1}(t)$ is defined as $\tilde{r}_1(t)$ reflected about $\tilde{r}_0(t)$. Evaluating these integrals gives

$$\left[\frac{r^\delta}{\delta} \tilde{u}_0(t) + \frac{r^{\delta+2}}{\delta+2} k \right]_{\tilde{r}_{-1}(t)}^{\tilde{r}_1(t)} \quad = \quad \left[\frac{r^\delta}{\delta} \tilde{u}_0(0) + \frac{r^\delta + 2}{\delta+2} k \right]_{\tilde{r}_{-1}(0)}^{\tilde{r}_1(0)}.$$

Since $\tilde{r}_{-1}(t) = -\tilde{r}_1(t)$ by symmetry,

$$\frac{\tilde{r}_1(t)^\delta}{\delta}\tilde{u}_0(t) + \frac{\tilde{r}_1(t)^{\delta+2}}{\delta+2}k \quad = \quad \frac{\tilde{r}_1(0)^\delta}{\delta}\tilde{u}_0(0) + \frac{\tilde{r}_1(0)^{\delta+2}}{\delta+2}k.$$

We assume that $\tilde{r}_1(t)k << 1$. Then

$$\tilde{u}_0(t) \quad \approx \quad \frac{\tilde{r}_1(0)^\delta}{\tilde{r}_1(t)^\delta}\tilde{u}_0(0),$$

which gives

$$u_0^{m+1} = \frac{(r^\delta)_1^0}{(r^\delta)_1^{m+1}}u_0^0.$$

The solution at the boundary, u_N^{m+1}, is determined by the boundary conditions (4.50).

4.6.3 The full algorithm

Given a mesh along a radius r_j^m, solution u_j^m, $j = 0, \ldots, N$, $m \geq 0$:

- Compute the mesh velocity s_j^m from (4.52) or (4.53);
- Compute the updated mesh r_j^{m+1} by a time-stepping scheme;
- Compute the updated solution u_j^{m+1} from (4.55) or (4.56).

In the last two sections we have given the details of applying the moving mesh method to the PME. In the next section we present the numerical results.

4.7 Numerical results

In this section we present results from applying the moving mesh method of §3.1 to the one-dimensional Cartesian case §4.5, and the two-dimensional radial case §4.6 with $\delta = 2$. We use the self-similar initial conditions from §4.4 to show that the numerical solution converges to the exact solution as the number of nodes increases in both the one-dimensional case (with explicit and semi-implicit time-stepping schemes), and the radial case. We also examine the effect of changing n, and using initial conditions that are not self-similar solutions of the PME.

All numerical results presented here start with a equispaced mesh and use a second order approximation (equation (4.38) in the one-dimensional case and equation (4.53) in the two-dimensional case) to calculate the mesh velocity since they use more information to obtain the derivative at a node. However, we also examined the numerical solution using (4.34) in the one-dimensional case and it was noted that in the tests they gave the same results as using (4.38) to at least $\mathcal{O}(10^{-12})$. Similarly, we used (3.16) to recover the solution, since it is more accurate for a non-uniform mesh. All the same, we examined the numerical solution using (3.10) and found that in the tests, the numerical solutions were the same to $\mathcal{O}(10^{-11})$. The minimal difference between these approaches suggests that the mesh remains fairly uniform when used to numerically solve the PME. Despite this, we use the approximations for a non-uniform mesh since in general, moving mesh methods often exhibit irregularity.

4.7.1 One-dimension

We begin by examining the convergence of the finite difference moving mesh method as the number of nodes N increases and as Δt decreases. We solve for $t \in [1,5]$ and compute results for $N = 10 \times 2^{\hat{N}-1}$, $\hat{N} = 1, \ldots, 6$. In order to compare results for different values of $\hat{N}$, we denote the points of the mesh for a particular value of $\hat{N}$ by $x_{j,\hat{N}}$, $j = 0, \ldots, (10 \times 2^{\hat{N}-1})$. We then compute both $x_{2^{\hat{N}-1}i,\hat{N}}$ and $u_{2^{\hat{N}-1}i,\hat{N}} \approx u(x_{2^{\hat{N}-1}i,\hat{N}}, 5)$ for each $i = 0, \ldots, 10$ as $\hat{N}$ increases. This new notation gives an approximation to the value of $\tilde{x}_j(5)$ and $\tilde{u}_j(5)$ at ten different points for various N determined by $j = 2^{\hat{N}-1}i$. We compare the numerical outcomes with the exact solution and boundary position from the one-dimensional Cartesian form of (4.26) and (4.25), with $n = 1$ and $t = 5$. To do this we introduce

$$\begin{aligned}\bar{u}_{2^{\hat{N}-1}i,\hat{N}} &= \frac{1}{5^{1/3}}\left(1 - \frac{(x_{2^{\hat{N}-1}i,\hat{N}})^2}{5^{4/3}6}\right),\\ \bar{x}_{N,\hat{N}} &= 5^{2/6}\sqrt{6},\end{aligned}$$

where $\bar{u}_{2^{\hat{N}-1}i,\hat{N}}$ is the exact solution at the calculated mesh points, and $\bar{x}_N$ is the exact boundary position, at $t = 5$. We use the corresponding initial conditions (4.27) and (4.28) such that

$$n = 1: \qquad \tilde{u}_j(1) = 1 - \frac{\tilde{x}_j(1)^2}{6}, \qquad \tilde{x}_N(1) = \sqrt{6}. \tag{4.57}$$

To balance the spatial and temporal errors, and recalling that we have mainly used explicit Euler time-stepping, we use $\Delta t = \mathcal{O}\left(\frac{1}{N^2}\right)$, precisely $\Delta t = \frac{4}{10(4^{\hat{N}})}$. We anticipate that the pointwise errors $|\bar{u}_{2^{\hat{N}-1}i,\hat{N}} - u_{2^{\hat{N}-1}i,\hat{N}}|$ and $|\bar{x}_{N,\hat{N}} - x_{N,\hat{N}}|$ will decrease as $\hat{N}$ increases, for each $i = 0, \ldots, 10$.

As a measure of the errors, we calculate ℓ_2 norms of u_j and the maximum normal of

x_N,

$$E_N(u) = \sqrt{\frac{\sum_{i=0}^{10}(\bar{u}_{2^{\hat{N}-1}i,\hat{N}} - u_{2^{\hat{N}-1}i,\hat{N}})^2}{\sum_{i=0}^{10}(\bar{u}_{2^{\hat{N}-1}i,\hat{N}})^2}}, \qquad E_N(x_N) = \frac{(\bar{x}_{N,\hat{N}} - x_{N,\hat{N}})}{(\bar{x}_{N,\hat{N}})}, \tag{4.58}$$

for $\hat{N} = 1, \ldots, 6$ (i.e. $N = 10, 20, 40, 80, 160, 320$). We investigate the hypothesis that

$$E_N(u) \sim \frac{1}{N^p} \qquad \text{and} \qquad E_N(x_N) \sim \frac{1}{N^q}, \tag{4.59}$$

for large N, where p and q are the estimated orders of convergence. If (4.59) holds then we would expect that p_{2N} and q_{2N} defined by

$$p_{2N} = -\log_2\left(\frac{E_{2N}(u)}{E_N(u)}\right), \qquad q_{2N} = -\log_2\left(\frac{E_{2N}(x_N)}{E_N(x)}\right),$$

would approach the constant values p and q as $N \to \infty$. Since each step of our scheme is second order in space and first order in time, and recalling that $\Delta t = \mathcal{O}\left(\frac{1}{N^2}\right)$, we might expect to see $p, q \approx 2$.

N	$E_N(u)$	p_N	$E_N(x_N)$	q_N
10	7.715×10^{-3}	-	1.451×10^{-3}	-
20	1.941×10^{-3}	2.0	3.066×10^{-4}	2.2
40	4.976×10^{-4}	2.0	7.138×10^{-5}	2.1
80	1.259×10^{-4}	2.0	1.730×10^{-5}	2.0
160	3.166×10^{-5}	2.0	4.262×10^{-6}	2.0
320	7.937×10^{-6}	2.0	1.058×10^{-6}	2.0

Table 4.1: Relative errors for u and x_N with rates of convergence using the explicit Euler time-stepping scheme, for $n = 1$.

Convergence results are shown in Table 4.1. We see that $E_N(u)$ and $E_N(x_N)$ decrease as N increases for each of the moving mesh methods. This suggests that as the number of nodes increases, both the solution $\tilde{u}_j(t)$ and the boundary position $\tilde{x}_N(t)$ are converging. The p and q-values presented strongly indicate second-order convergence of both the numerical solution and numerical boundary position.

Having shown apparent convergence of our moving mesh schemes, we examine the numerical results with $N = 20$. We consider $n = 1, 2, 3$ with self-similar initial conditions given by the one-dimensional Cartesian case of (4.27) and (4.28). The $n = 1$ case has initial conditions given by (4.57) and the $n = 2, 3$ cases are given by

$$n = 2: \qquad \tilde{u}_j(1) = \left(1 - \frac{\tilde{x}_j(1)^2}{4}\right)^{\frac{1}{2}}, \qquad \tilde{x}_{20}(1) = 2, \tag{4.60}$$

$$n = 3: \qquad \tilde{u}_j(1) = \left(1 - \frac{3\tilde{x}_j(1)^2}{10}\right)^{\frac{1}{3}}, \qquad \tilde{x}_{20}(1) = \sqrt{\frac{10}{3}}. \tag{4.61}$$

The results from self-similar initial conditions are given in Figures 4.3–4.5. We have also applied (symmetric) initial conditions that are not self-similar for $n = 2, 3$,

$$n = 2: \qquad \tilde{u}_j(1) = 1 - \frac{\tilde{x}_j(1)^2}{4}, \qquad \tilde{x}_{20}(1) = 2, \tag{4.62}$$

$$n = 3: \qquad \tilde{u}_j(1) = 1 - \frac{3\tilde{x}_j(1)^2}{10}, \qquad \tilde{x}_{20}(1) = \sqrt{\frac{10}{3}}, \tag{4.63}$$

from removing the square roots in (4.60) and (4.61). The results from these initial conditions are given in Figures 4.6 and 4.7. Due to symmetry, the moving mesh method was applied to half the region $x \in [0, b(t)]$. The computed outcome (using the explicit Euler time-stepping scheme) is used to provide a complete numerical solution over the region $[-\tilde{x}_{20}(t), \tilde{x}_{20}(t)] \approx [a(t), b(t)]$, as shown in the first part of Figures 4.3–4.7.

When a self-similar solution is used for the initial conditions (4.57), (4.60) and (4.61), we have an exact solution to compare with our numerical outcomes. We define the exact boundary position by the one-dimensional Cartesian case of (4.25),

$$\bar{x}_{n,N}(t) = \sqrt{\frac{2(n+2)t^{\frac{2}{n+2}}}{n}},$$

and the exact solution at the position of the nodes by the one-dimensional Cartesian case of (4.26)

$$\bar{u}_{n,j}(t) = \frac{1}{t^{\frac{1}{2+n}}}\left(1 - \frac{\tilde{x}_j(t)^2}{\tilde{x}_N(0)^2}\right)^{\frac{1}{n}}. \tag{4.64}$$

Figure 4.8 shows the difference

$$d_{n,j}(t) = \bar{u}_{n,j}(t) - \tilde{u}_j(t),$$

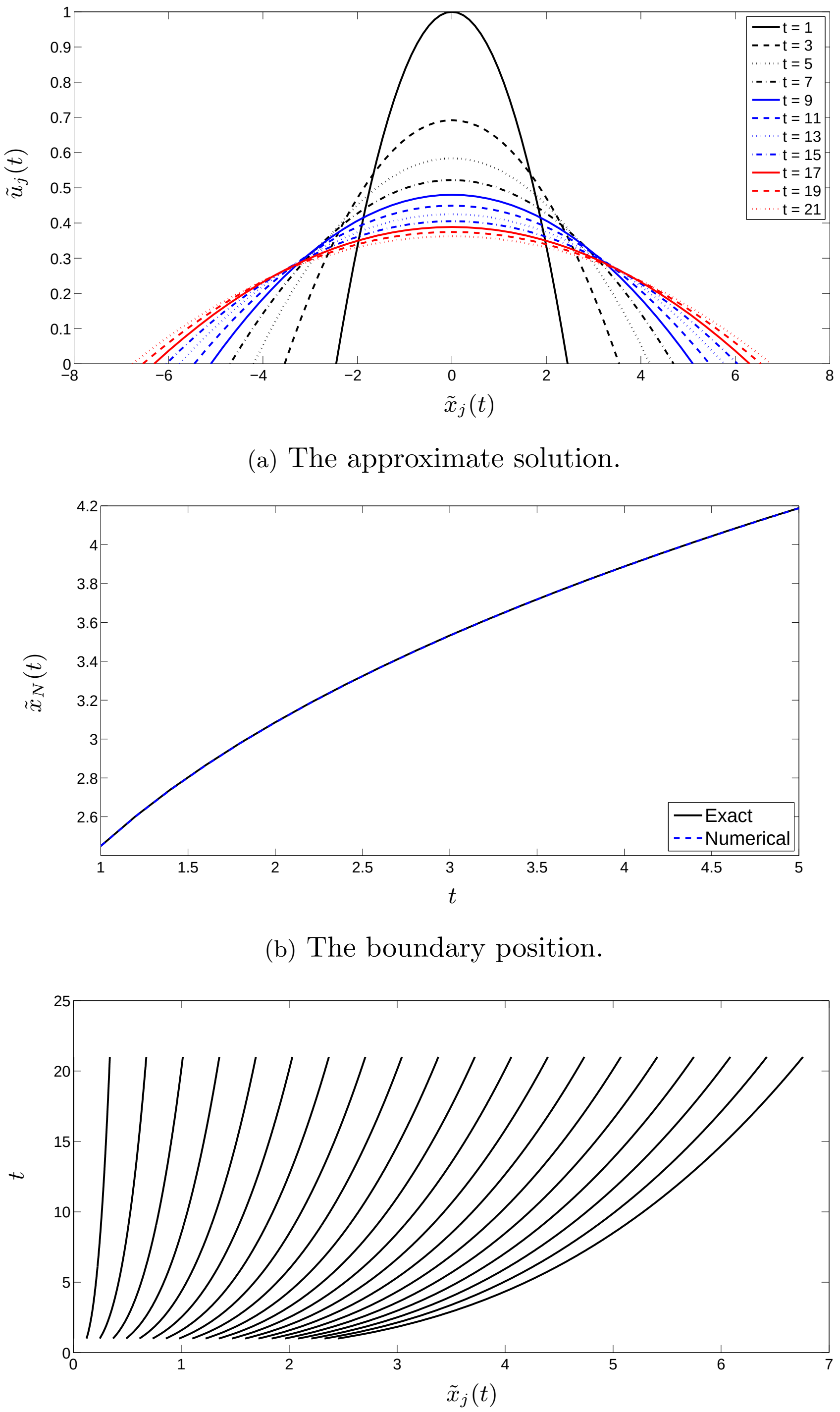
at each node, $j = 0, 1, \ldots, 20$, between the exact and numerical solution for $n = 1, 2, 3$ at $t = 5$. The difference appears to increase near the boundary (but the difference $d_{n,20}(t)$=0 is zero due to known boundary conditions). The cause of the greater difference near the outer boundary may be that the boundary mesh velocity is determined by an extrapolation of the internal nodes. Moreover, from (4.64) we note that the exact solution for $n = 2, 3$

has an infinite gradient at the boundaries. This is portrayed in Figures 4.4(a) and 4.5(a) which show the gradient of the solution near the moving boundary is very large for $n = 2, 3$, resulting in a less accurate approximation near the moving boundary than for the $n = 1$ case, shown in Figure 4.3. Hence, from Figure 4.8, it is reasonable to deduce that a good graphical indication of the accuracy of our method is to compare the exact and numerical outer boundary movement, as shown in the second plot of Figures 4.3–4.5. These plots show that the exact boundary position is slightly larger than the numerical boundary position, with the difference between the two increasing as n increases. Returning to Figure 4.8 we observe that for $n = 1$ the largest positive error is near the inner boundary. Subsequently, we present the difference between the exact and numerical solution at the inner boundary $d_{n,0}(t)$, see Figure 4.9. This shows that for short times $t < 2$, the difference $d_{n,0}(t)$ is larger for smaller n. However, as t increases, $d_{1,0}(t)$ decreases, whereas $d_{2,0}(t)$, $d_{3,0}(t)$ increase, with $d_{3,0}(t)$ increasing at a faster rate than $d_{2,0}(t)$. The varying behaviour for different n is because the solution decreases in the centre (the inner boundary) at a faster rate for smaller n. The relative error at the inner boundary would present less variation for different n.

The boundary movement for non-self-similar initial conditions is also presented, Figures 4.6(b) and 4.7(b), but we do not have exact solutions to enable a comparison. These plots are included to demonstrate ‘waiting times’ as discussed in §4.5.4. For the initial conditions (4.62) and (4.63) we observe that $\alpha = 1$ in both cases, giving $\alpha n > 1$ for $n = 2, 3$. A waiting time is nicely demonstrated in the $n = 2$ case, Figure 4.6. In addition, the velocity of the nodes is shown in Figure 4.10, where we observe that the boundary velocity is initially zero. A waiting time is also visible in the $n = 3$ case. However, the boundary moves in slightly before moving out, as shown in Figure 4.7. This inward movement may be due to numerical errors caused by extrapolating the boundary mesh velocity v_N^m.

The third plot from Figures 4.3–4.7 shows exactly how the mesh moves. We notice a smooth even spread of the nodes, without any tangling in all five cases. The two cases where the initial conditions are not self-similar, (4.62) and (4.63), are more challenging for our moving mesh method since there is a ‘waiting time’; however, our method produces a mesh that is finer near the boundary, which is where the gradient of the solution is at its highest.

Using a semi-implicit time-stepping scheme allows a larger Δt compared to the explicit Euler time-stepping scheme, see Figure 4.11. Although the semi-implicit scheme can take very large time steps, such as $\Delta t = 20$, without mesh tangling, it does not ensure accuracy since both plots of Figure 4.11 show the solution at $t = 20$ but the solutions vary considerably.

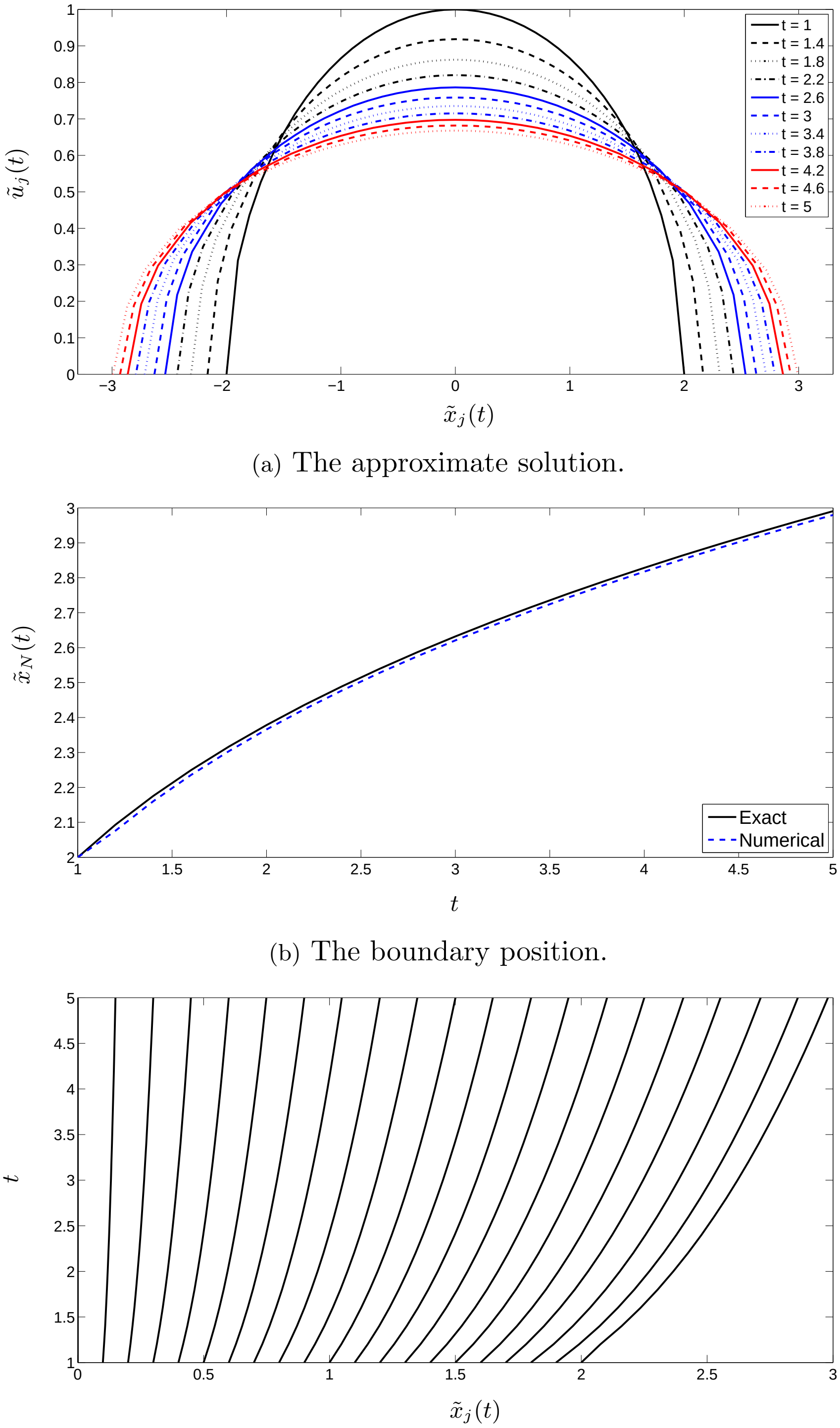


(a) The approximate solution.

(b) The boundary position.

(c) The mesh trajectory.

Fig. 4.3: The PME with self-similar initial conditions for $n = 1$ (4.57), $N = 20$, $\Delta t = 0.04$.

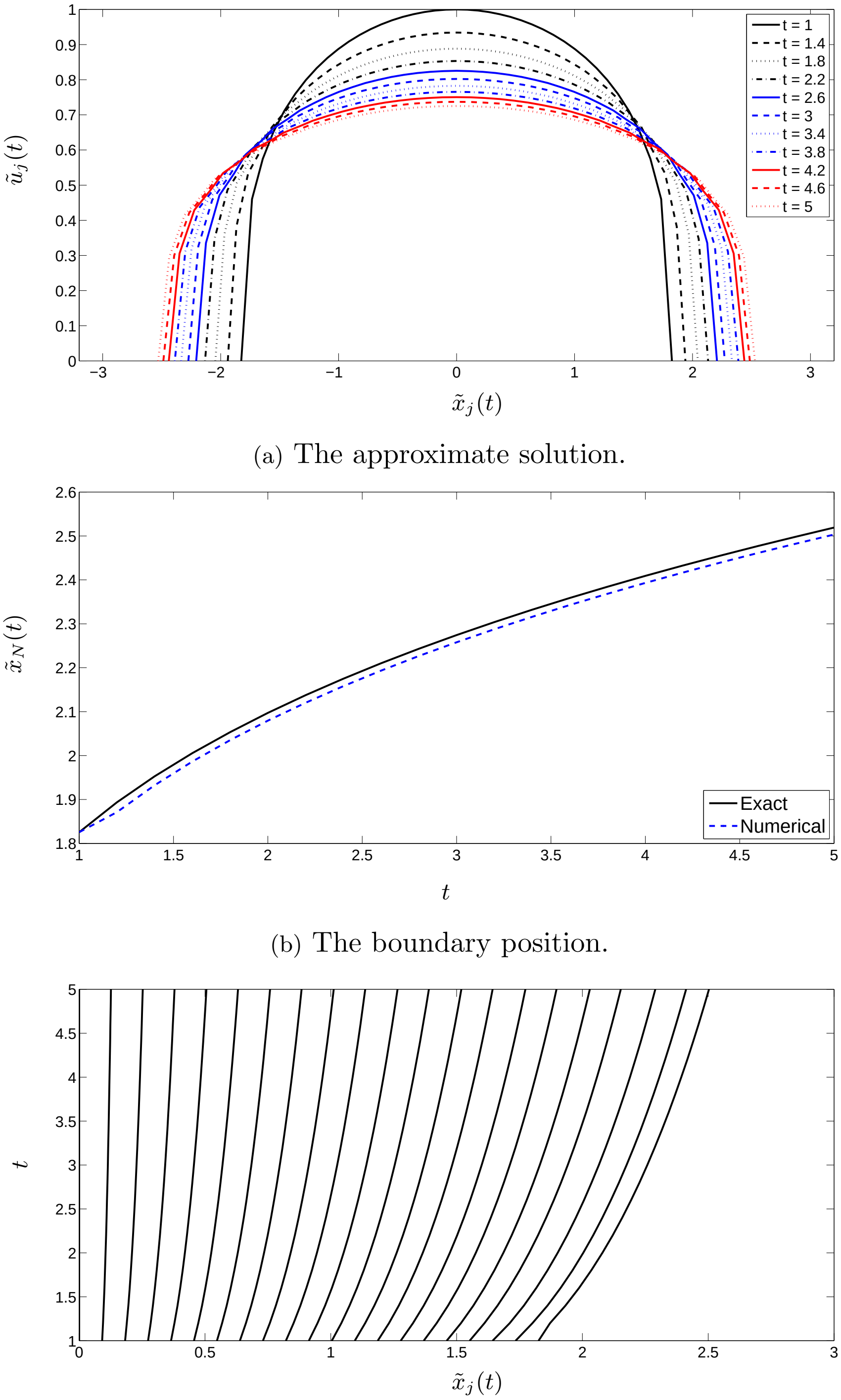


(a) The approximate solution.

(b) The boundary position.

(c) The mesh trajectory.

Fig. 4.4: The PME with self-similar initial conditions for $n = 2$ (4.60), $N = 20$, $\Delta t = 0.04$.

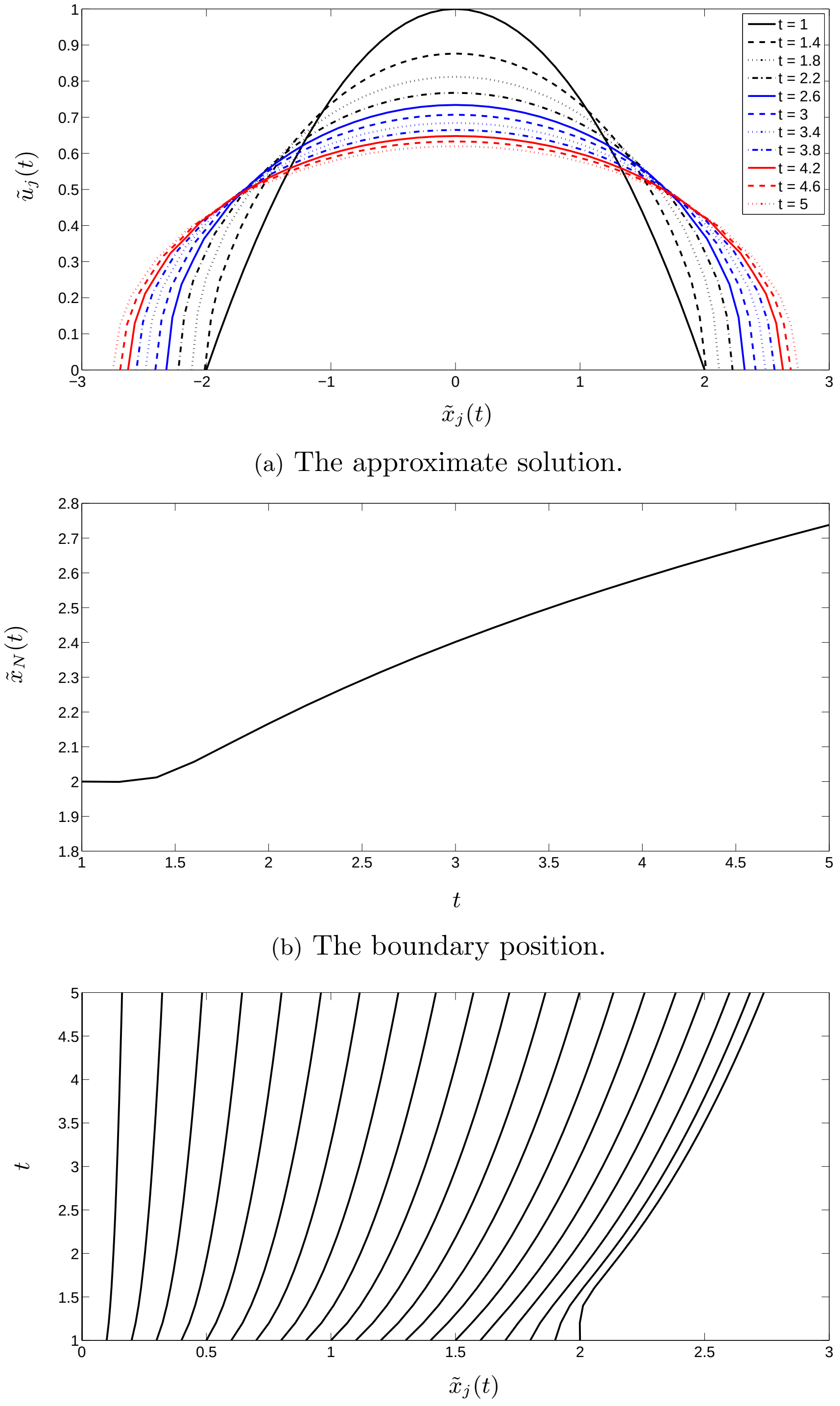


(a) The approximate solution.

(b) The boundary position.

(c) The mesh trajectory.

Fig. 4.5: The PME with self-similar initial conditions for $n = 3$ (4.61), $N = 20$, $\Delta t = 0.04$.

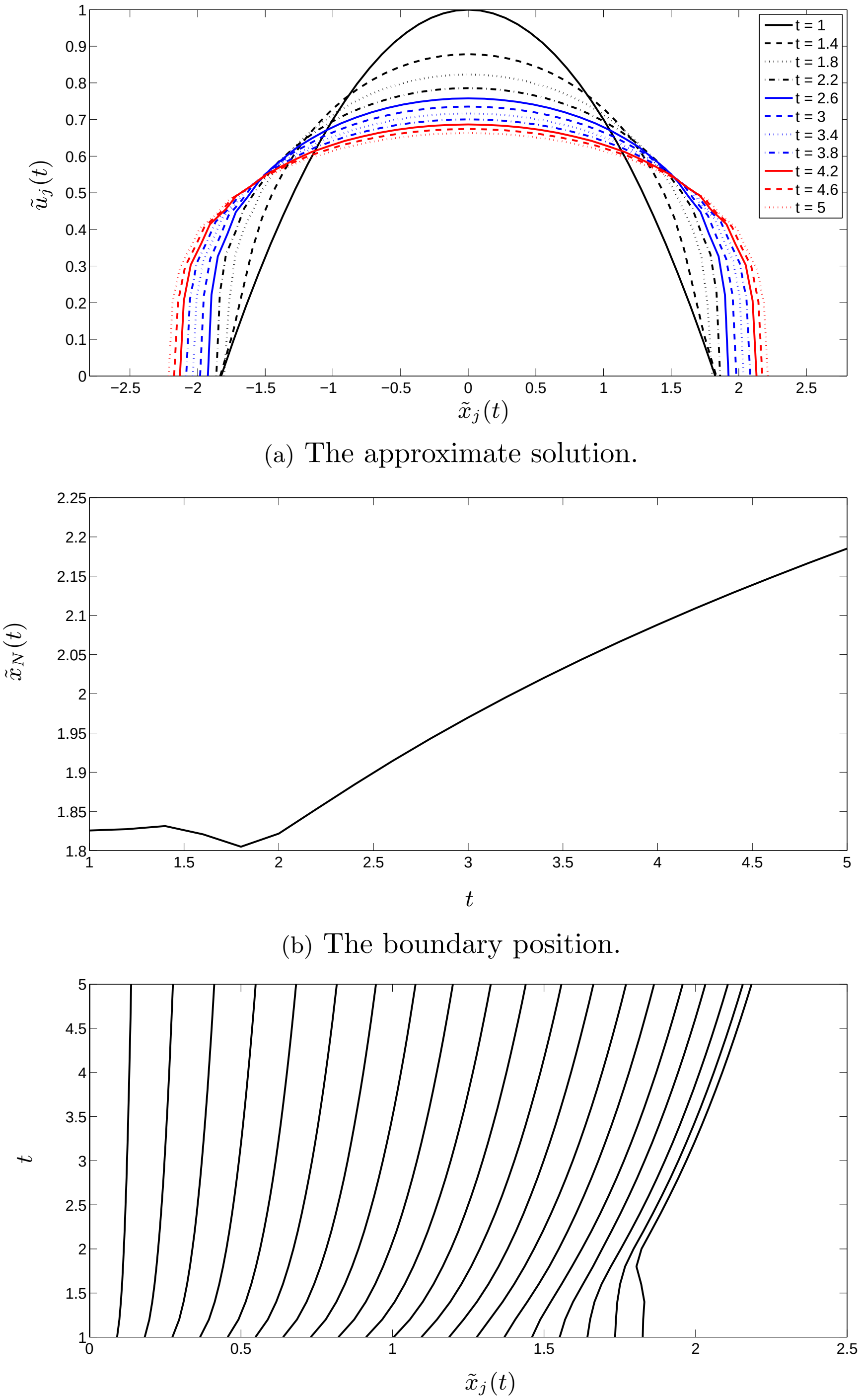


(a) The approximate solution.

(b) The boundary position.

(c) The mesh trajectory.

Fig. 4.6: The PME without self-similar initial conditions for $n = 2$ (4.62), $N = 20$, $\Delta t = 0.01$.



(a) The approximate solution.

(b) The boundary position.

(c) The mesh trajectory.

Fig. 4.7: The PME without self-similar initial conditions for $n = 3$ (4.62), $N = 20$, $\Delta t = 0.01$.

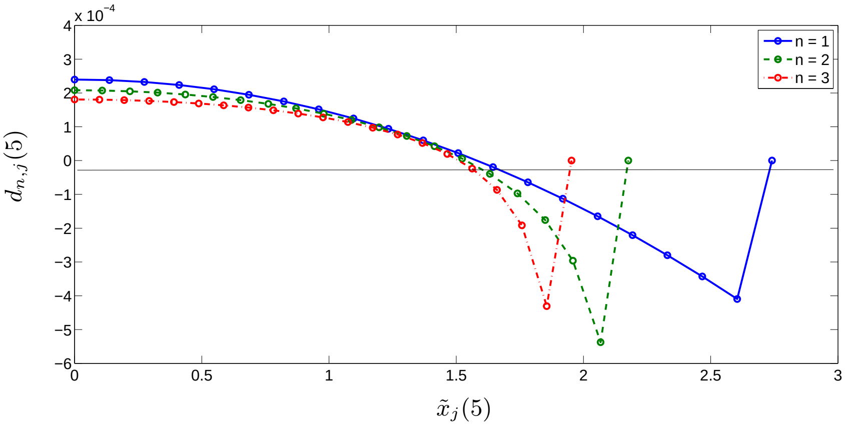


Fig. 4.8: The difference of the PME with $n = 1, 2, 3$ and corresponding self-similar solution (4.57)–(4.61) at each node, $N = 20$, $t = 5$, $\Delta t = 0.01$.

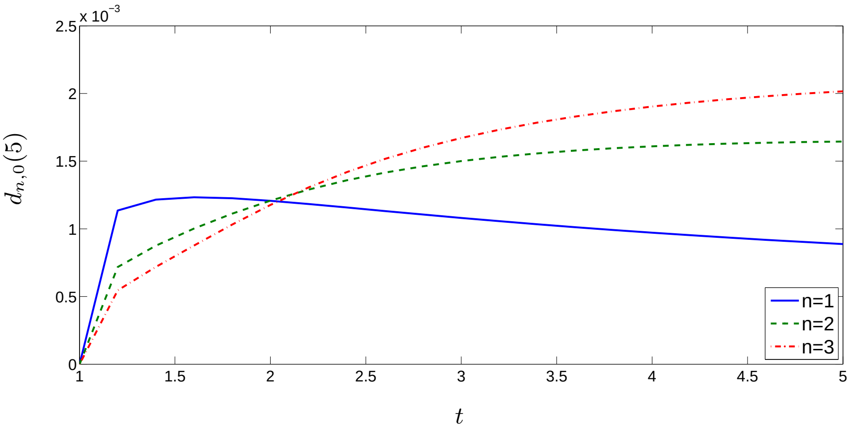


Fig. 4.9: The difference of the PME with corresponding self-similar solution (4.57)–(4.61) at the inner boundary $\tilde{x}_0(t)$, $N = 20$, $\Delta t = 0.01$.

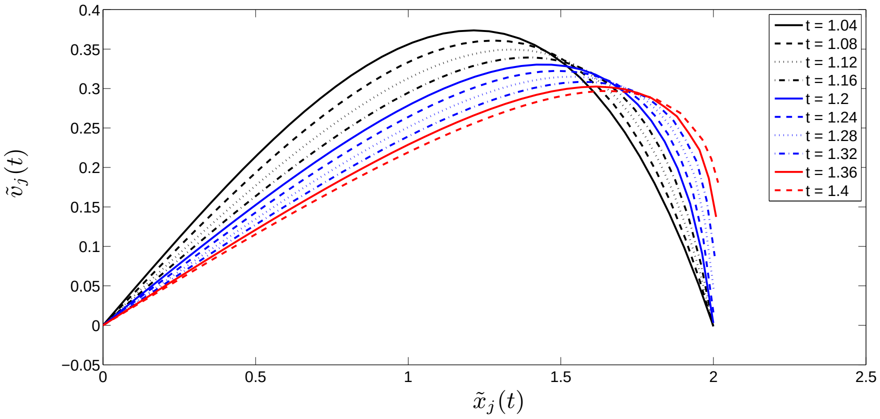


Fig. 4.10: The velocity of the nodes without self-similar initial conditions for $n = 2$ (4.62), $N = 40$, $\Delta t = 0.01$.

4.7.2 Two-dimensional radially symmetric

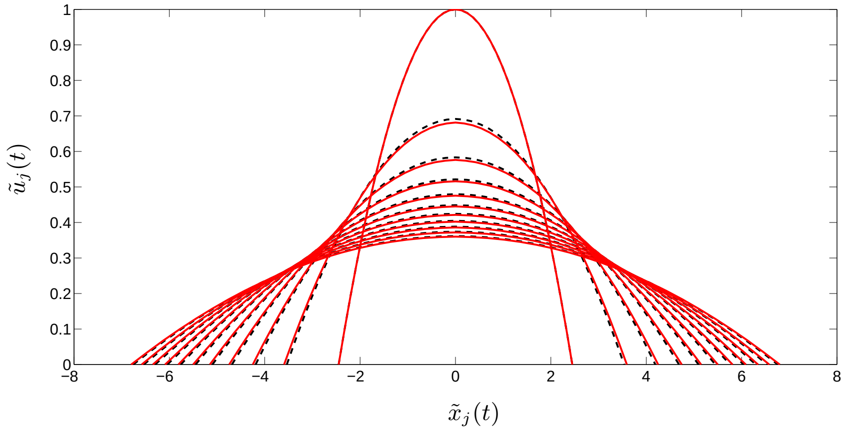
As with the one-dimensional case, we begin by examining the convergence as the number of nodes N increases and as Δt decreases. The moving mesh method for the symmetrical radial case solves the PME along a line $0 \leq r \leq r_N$, and then uses radial symmetry to give the solution over a complete circle, see Figure 4.12. To examine convergence of this method we take the solution along the line $0 \leq r \leq r_N$, and compare this to the exact solution.

We use the initial conditions given by (4.27) and (4.28) with $\delta = 2$ and $t = 1$,

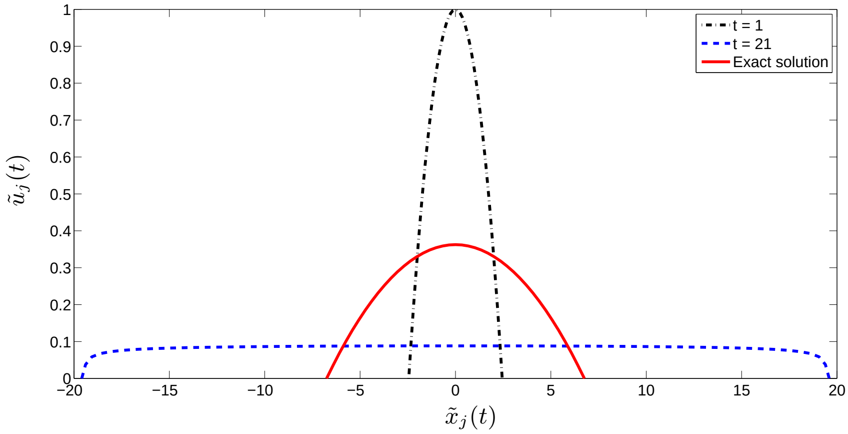
$$n = 1: \qquad \tilde{u}_j(1) = 1 - \frac{\tilde{r}_j(1)^2}{8}, \qquad \tilde{r}_N(1) = \sqrt{8}. \tag{4.65}$$

We use the same parameters as in the one-dimensional case, and so we solve for $t \in [1, 5]$ and compute results for $N = 10 \times 2^{\hat{N}-1}$, $\hat{N} = 1, \ldots, 6$. In order to compare results for different values of $\hat{N}$, we denote the mesh points along the line $0 \leq r \leq r_N$, for a particular value of $\hat{N}$, by $r_{j,\hat{N}}$, $j = 0, \ldots, (10 \times 2^{\hat{N}-1})$. We then compute both $r_{2^{\hat{N}-1}i,\hat{N}}$ and $u_{2^{\hat{N}-1}i,\hat{N}} \approx u(r_{2^{\hat{N}-1}i,\hat{N}}, 5)$ for each $i = 0, \ldots, 10$ as $\hat{N}$ increases. This notation gives an approximation to the value of $\tilde{r}_j(5)$ and $\tilde{u}_j(5)$ at ten different points for various N determined by $j = 2^{\hat{N}-1}i$. We compare the numerical outcomes with the exact solution and boundary position from (4.26) and (4.25) where $\delta = 2$, $n = 1$ and $t = 5$,

$$\begin{aligned} \bar{u}_{2^{\hat{N}-1}i,\hat{N}} &= \frac{1}{5^{1/4}}\left(1 - \frac{(r_{2^{\hat{N}-1}i,\hat{N}})^2}{40}\right), \\ \bar{r}_N &= 5^{1/4}\sqrt{8}, \end{aligned}$$



(a) Taking $\Delta t = 0.2$ (solid red). For comparison, the explicit Euler time-stepping scheme with $\Delta t = 0.4$ (as in Figure 4.3(a)) is also plotted (dashed black).



(b) Taking $\Delta t = 20$.

Fig. 4.11: The PME with self-similar initial conditions for $n = 1$ (4.57), using a semi-implicit time-stepping scheme, $N = 20$.

where $\bar{u}_{2^{\hat{N}-1}i,\hat{N}}$ is the exact solution at the calculated mesh points, and $\bar{r}_N$ is the radius, at $t = 5$. As before, to balance the spatial and temporal errors, and recalling that we have used explicit Euler time-stepping, we use $\Delta t = \mathcal{O}\left(\frac{1}{N^2}\right)$, precisely $\Delta t = \frac{4}{10(4^{\hat{N}})}$. We anticipate that the pointwise errors $|\bar{u}_{2^{\hat{N}-1}i,\hat{N}} - u_{2^{\hat{N}-1}i,\hat{N}}|$ and $|\bar{r}_{N,\hat{N}} - r_{N,\hat{N}}|$ will decrease as $\hat{N}$ increases, for each $i = 0, \ldots, 10$.

As a measure of the errors, we calculate the first part of (4.58) and

$$E_N(r_N) = \frac{(\bar{r}_{N,\hat{N}} - r_{N,\hat{N}})}{(\bar{r}_{N,\hat{N}})},$$

for $\hat{N} = 1, \ldots, 6$ (i.e. $N = 10, 20, 40, 80, 160, 320$). We investigate the same hypothesis as with the one-dimensional case: that

$$E_N(u) \sim \frac{1}{N^p} \qquad \text{and} \qquad E_N(r_N) \sim \frac{1}{N^q}, \tag{4.66}$$

for large N, where p and q are the estimated orders of convergence. If (4.66) holds then we would expect that p_{2N} and q_{2N} defined by

$$p_{2N} = -\log_2\left(\frac{E_{2N}(u)}{E_N(u)}\right), \qquad q_{2N} = -\log_2\left(\frac{E_{2N}(r_N)}{E_N(r)}\right),$$

would approach the constant values p and q as $N \to \infty$. Since each step of our scheme is second order in space and first order in time, and recalling that $\Delta t = \mathcal{O}\left(\frac{1}{N^2}\right)$, we might expect to see $p, q \approx 2$. Convergence results are shown in Table 4.2, where we observe very similar results to the one-dimensional case in Table 4.1. We find that the $\tilde{u}_j(t)$ and the boundary position $\tilde{r}_N(t)$ appear to have second-order convergence.

N	$E_N(u)$	p_N	$E_N(r_N)$	q_N
10	5.519×10^{-3}	-	3.072×10^{-3}	-
20	1.364×10^{-3}	2.0	7.577×10^{-4}	2.0
40	3.401×10^{-4}	2.0	1.888×10^{-4}	2.0
80	8.497×10^{-5}	2.0	4.716×10^{-5}	2.0
160	2.124×10^{-5}	2.0	1.179×10^{-6}	2.0
320	5.309×10^{-6}	2.0	2.947×10^{-6}	2.0

Table 4.2: Relative errors for u and r_N with rates of convergence using the explicit Euler time-stepping scheme.

Having shown apparent convergence of the radial moving mesh method we present numerical results, with $N = 20$. We consider $n = 1, 2, 3$ with self-similar initial conditions given by (4.26) with $\delta = 2$ and $t = 1$, i.e. the $n = 1$ case is given by (4.65) and the $n = 2, 3$

cases are given by

$$n = 2: \qquad \tilde{u}_j(1) = \left(1 - \frac{\tilde{r}_j(1)^2}{6}\right)^{\frac{1}{2}}, \qquad \tilde{x}_{20}(1) = \sqrt{6}, \tag{4.67}$$

$$n = 3: \qquad \tilde{u}_j(1) = \left(1 - \frac{3\tilde{r}_j(1)^2}{16}\right)^{\frac{1}{3}}, \qquad \tilde{x}_{20}(1) = \frac{4}{\sqrt{3}}. \tag{4.68}$$

The Figure 4.12 confirm the findings from the one-dimensional case, that as n increases, the boundary slope is closer to infinity. Note that for $n = 3$ a smaller Δt was required. Although it appears that the mesh is coarser in the radial direction at the boundary, we observe from Figure 4.13 that this is not actually the case.

We have completed our application of our finite difference moving mesh method applied to the PME. In the next section we present our results from applying the finite element moving mesh method of Baines, Hubbard and Jimack [5] to the PME.

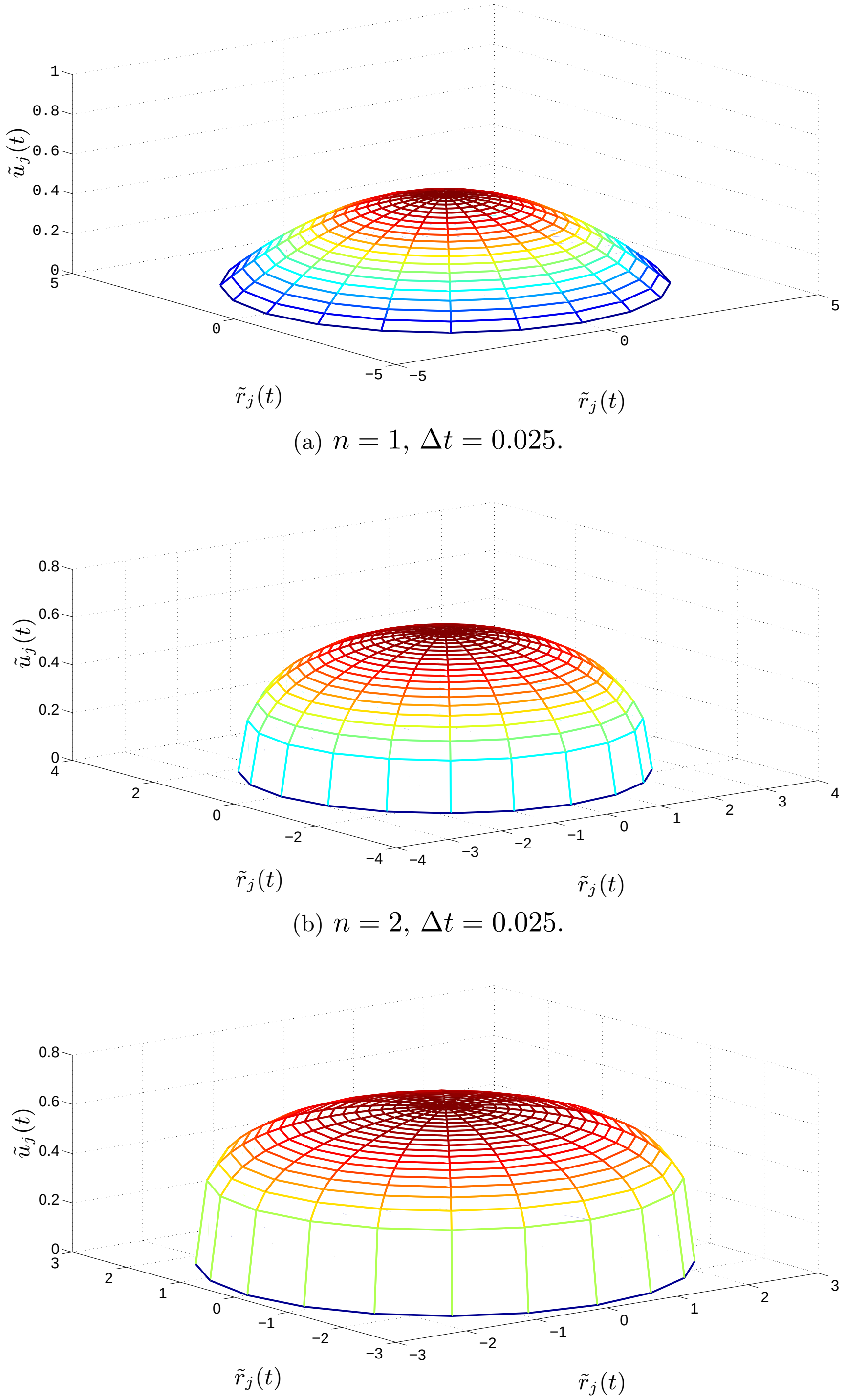
4.8 Finite elements

As mentioned in §2.3, the moving mesh method we use is a one-dimensional finite difference version of the multi-dimensional Conservation Method given in [5], which uses linear finite elements. For completeness we also solved the PME on a circle using the Conservation Method with linear finite elements, although with a slight alteration in the code. The numerical procedure is as given in §2.3.

Unlike [5], the mesh here is reconstructed at each time step by constructing a Delaunay triangulation from a set of N points. Newer versions of Matlab have a in-built Matlab function (‘DelaunayTri’) that computes the Delaunay triangulation for a set of given points, see Figure 4.14. For N points we have L triangles.

We consider the $n = 1$ case with a self-similar initial condition, and use the explicit Euler time-stepping scheme. The numerical solution is shown in Figure 4.15. We observe from Figure 4.15(b) that the mesh is finer at the centre. This has partly been overcome by spreading the nodes along a radius non-uniformly so that they are spread further apart at the centre. Generally, a mesh that is equally spaced is preferable, although this mesh is suitable for radial movement.

An advantage of using finite elements for the general two-dimensional case (instead of the radial case) is that the method can be applied to more general domains, such as an ellipse, as seen in Figure 4.16. For a general two-dimensional problem, an equally distributed mesh of equilateral triangles would be more suitable than our present choice of mesh.



(a) $n = 1$, $\Delta t = 0.025$.

(b) $n = 2$, $\Delta t = 0.025$.

(c) $n = 3$, $\Delta t = 0.0125$.

Fig. 4.12: The two-dimensional, radial PME with self-similar initial conditions for $n = 1$ (4.57), $n = 2$ (4.67) and $n = 3$ (4.68), at $t = 5$, using the explicit Euler time-stepping scheme, $N = 20$.

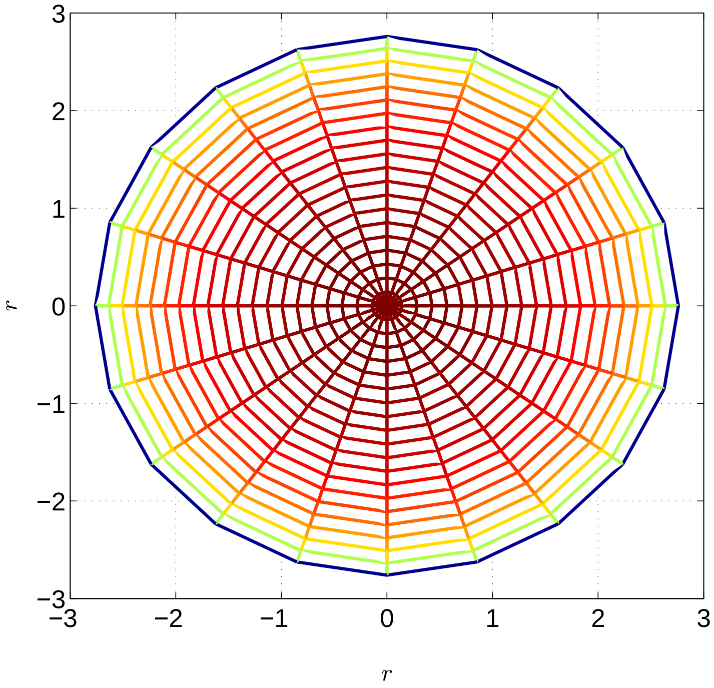


Fig. 4.13: The mesh for the two-dimensional, radial PME with self-similar initial conditions for $n = 3$ (4.68), $N = 20$, at $t = 5$, $\Delta t = 0.0125$

In the next section we summarise our work on the PME.

4.9 Summary for the PME

The PME is the first PDE we solved numerically with our moving mesh method since it is the simplest nonlinear diffusion equation that is of interest to both the pure mathematician and the applied scientist [99]. There are a number of physical applications where this simple model appears in a natural way, mainly to describe processes involving fluid flow, heat transfer or diffusion. We began this chapter with a derivation of the PME.

Budd, Huang and Russell [28] note that Lagrangian moving mesh methods, like our method, are very natural in fluid mechanics calculations (as in the PME) since solution features are often convected with the flow. Furthermore, it is natural to evolve the mesh points to follow the flow itself [28], which is what happens here since mass is conserved. We have discussed some of the properties, paying particular attention to the scaling properties which enabled us to derive a self-similar solution as originally shown in [11]. We used the

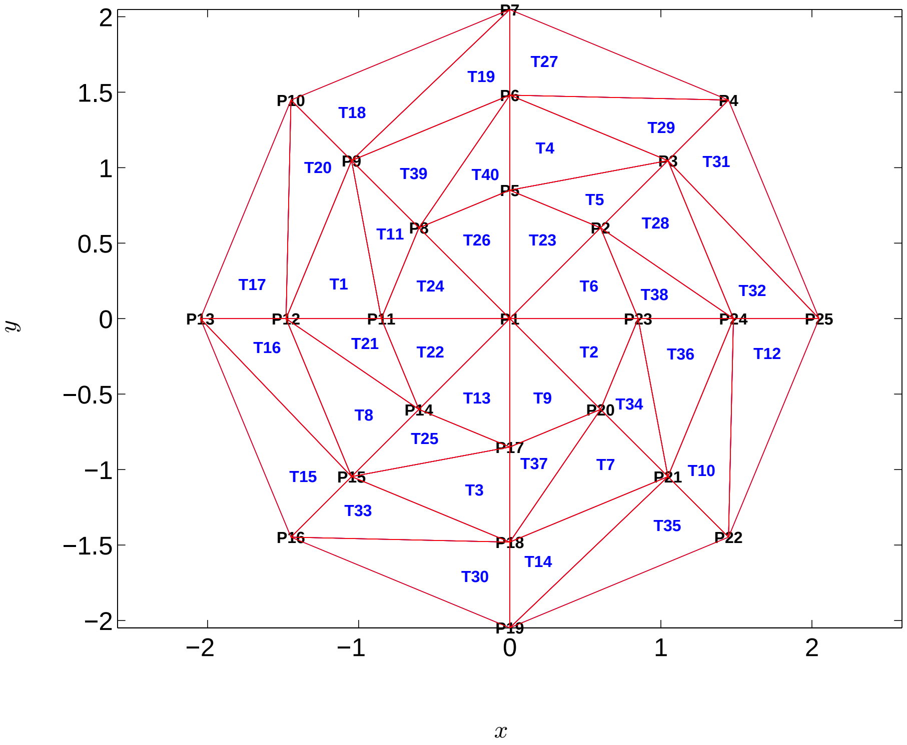
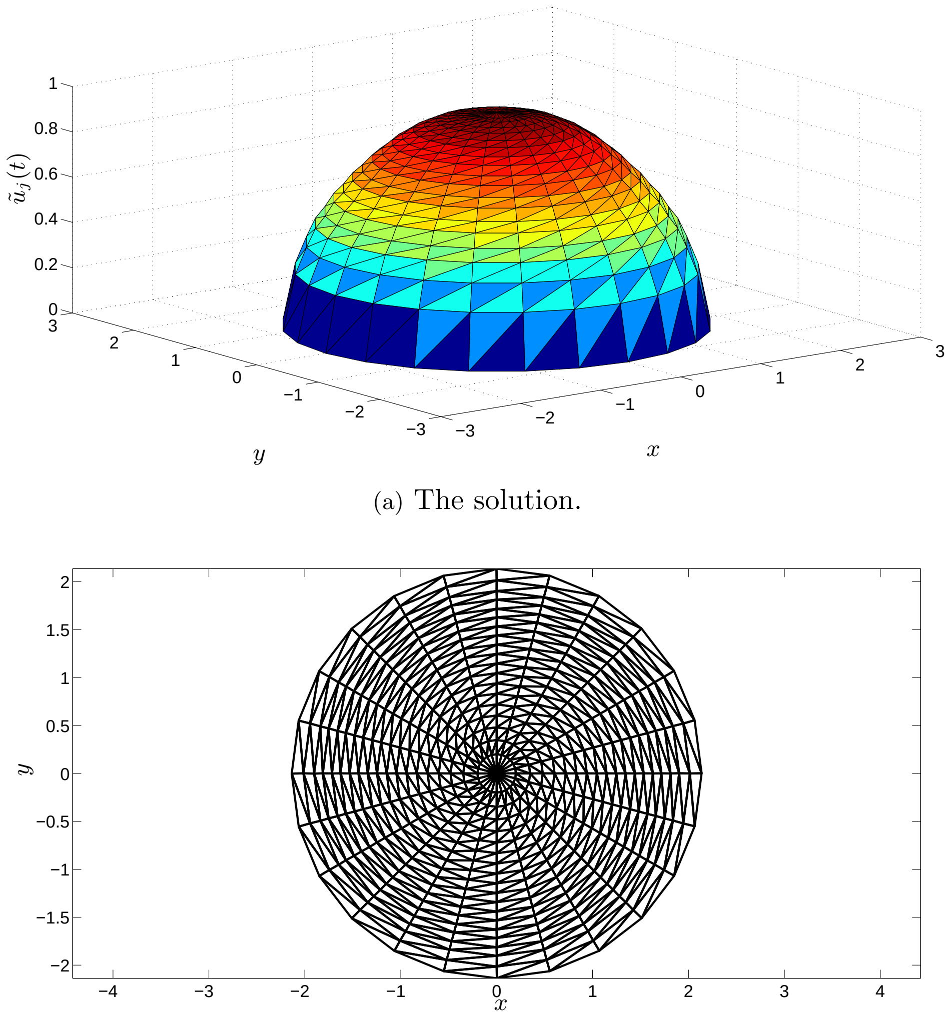


Fig. 4.14: Finite element mesh for a radial 2D problem, ($N = 25$, 40 triangles).



(a) The solution.

(b) The mesh.

Fig. 4.15: The numerical solution of the two-dimensional PME with initial conditions (4.65), $N = 20$, at $t = 2$, $\Delta t = 0.001$ using a finite element moving mesh.

self-similar solution as an initial condition so that we could compare our numerical results to the exact solution. We found that our moving mesh method is accurate, and the numerical solution and mesh appear to have second-order convergence when $n = 1$..

We also demonstrated that our moving mesh method can easily be extended to the radially symmetric two-dimensional case, without compromising accuracy, nor the rate of convergence. However, when using finite differences the two-dimensional case must be radially symmetric, which is not a limitation when using the Conservation Method with finite elements. We demonstrated this when using the Conservation Method of Baines, Hubbard

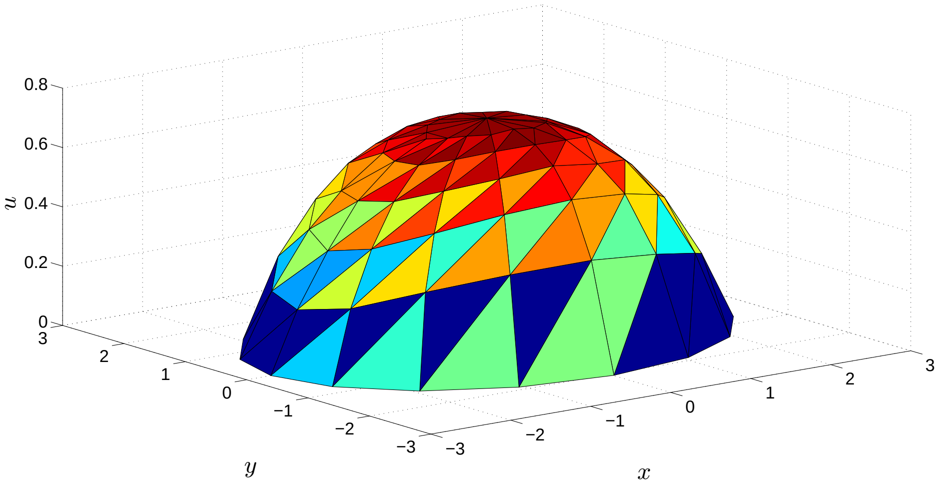


Fig. 4.16: The numerical solution of the two-dimensional PME with initial conditions (4.65), scaled such that $x \to 1.2x$ and $y \to 0.8y$, $N = 20$, at $t = 2$, $\Delta t = 0.02$ using a finite element moving mesh.

and Jimack [5].

We now consider Richards' equation, which is another fluid flow problem that conserves mass. However, unlike the PME, it does not remain symmetrical.

5 Richards' Equation

5.1 Introduction

Richards' equation is a non-linear PDE which models the movement of moisture in an unsaturated porous medium. It was formulated in 1931 by Richards' [83] in the form

$$\nabla \cdot (K \cdot \nabla \psi) + g\frac{\partial K}{\partial z} = \rho \frac{\mathrm{d}\theta}{\mathrm{d}\psi}\frac{\partial \psi}{\partial t}, \tag{5.1}$$

where θ is the moisture content, ψ is the liquid pressure, which is non-negative and unbounded as $\theta \to 0$, p is the mass per unit volume of the liquid, ρ is the weight of the dry medium in unit volume, g is a gravity constant, z is the vertical direction, $K(\psi)$ is the hydraulic conductivity and t is time.

For the case of one-dimensional infiltration of water in the vertical direction of unsaturated soil, Richards' equation is,

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial z}\left[K(\theta)\left(\frac{\partial \psi}{\partial z} + 1\right)\right],$$

as given in [10]. This is the mixed form of Richards' equation. In our work we focus on the θ-based form

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial z}\left[D(\theta)\frac{\partial \theta}{\partial z}\right] + \frac{\partial K}{\partial z},$$

where the diffusion coefficient

$$D(\theta) = -K(\theta)\frac{\mathrm{d}\psi}{\mathrm{d}\theta} \quad \text{and} \quad \psi \propto \frac{1}{\theta}.$$

This is the form of Richards' equation presented in [50], where the author goes on to assume that

$$K(\theta) \propto \theta^n,$$

for some integer $n > 2$. Rescaling z and t then gives the non-dimensional equation

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial z}\left[\theta^{n-2}\frac{\partial \theta}{\partial z} + \theta^n\right]. \tag{5.2}$$

This is a specific case of the θ-based form of Richards' equation, as given in [10, 61], which is defined only for unsaturated flow. Hence, the solution to (5.2) represents liquid flowing downwards through an unsaturated porous medium, making it applicable to track contaminated liquid seeping downwards through soil.

We alter the notation of (5.2) to be consistent with our description of the moving mesh method, and the notation in the rest of this thesis, such that the orientation is transformed from the z to the x-axis, and the moisture content is $u(x,t)$ instead of $\theta(x,t)$, giving

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(u(x,t)^{n-2}\frac{\partial u}{\partial x} + u(x,t)^n\right), \qquad x \in [a(t), b(t)], \tag{5.3}$$

where $n > 2$. We consider zero Dirichlet boundary conditions since the region we are modelling is defined by presence of the liquid,

$$u = 0 \text{ at } x = a(t), b(t). \tag{5.4}$$

We use an initial condition

$$u = u^0(x) \quad \text{at } t = 0,$$

which will be defined later. As with the PME, Richards' equation is parabolic everywhere except where $u = 0$, where it degenerates. In addition, like the PME, the boundaries have a finite propagation velocity. Richards' equation is closely related to the PME with some similar properties.

In the next section we show how Richards' [83] originally derived (5.1) from Darcy's Law to model the capillary conduction of liquids in a porous medium. We specialise the derivation presented in [83] to the one-dimensional case since we focus our work on the case where fluid is flowing in one direction only.

The second half of this chapter applies a moving mesh method. In §5.5 we use the mass conservation finite difference method, and update the mesh using both an explicit and semi-implicit time-stepping scheme. However, for Richards' equation, the mesh velocity from our mass conservation method cannot be arranged so that (3.39) holds, so the proof of monotonicity under semi-implicit time-stepping does not hold in this case. An alternative approach to define the mesh velocity (similar to that in §3.3) is explored so that (3.39) does hold. Numerical results from the finite difference methods are given in §5.7. In §5.8 the finite element method is applied to the one-dimensional case, corresponding to that presented in [90]. We give the results from the one-dimensional finite element method in §5.8.

5.2 Deriving Richards' equation

We describe the one-dimensional derivation of Richards' equation, derived from [83], where the equation of continuity and Darcy's Law are applied to the flow of liquid through a porous medium.

(i) Equation of continuity

The equation of continuity for fluid flow through a porous medium may be written as

$$\frac{\partial q}{\partial z} = -\rho \frac{\partial \theta}{\partial t}, \tag{5.5}$$

where q is the flow vector and θ is the moisture content. Since θ is a single-valued continuous function of the moisture tension (or pressure head) ψ, equation (5.5) becomes

$$\frac{\partial q}{\partial z} = -\rho \frac{\mathrm{d}\theta}{\mathrm{d}\psi} \frac{\partial \psi}{\partial t}. \tag{5.6}$$

(ii) Darcy's Law

Darcy's Law may be expressed by

$$q = -K \frac{\partial}{\partial z}(\phi + \psi),$$

where q is the volume of liquid crossing a unit area perpendicular to the flow, in unit time, and in this case ϕ is the potential $\phi = gz$. If the z-axis is chosen to be the positive upward vertical then $\frac{\partial \phi}{\partial z} = g$, so that

$$q = -K \left(g + \frac{\partial \psi}{\partial z} \right). \tag{5.7}$$

Substituting q from the Darcy's Law result (5.7) into the equation of continuity (5.6) gives

$$\frac{\partial K}{\partial z} \left(g + \frac{\partial \psi}{\partial z} \right) + K \left(\frac{\partial g}{\partial z} + \frac{\partial^2 \psi}{\partial z^2} \right) = \rho \frac{\mathrm{d}\theta}{\mathrm{d}\psi} \frac{\partial \psi}{\partial t}.$$

Dropping the $\frac{\partial g}{\partial z}$ term because g is presumed constant, and rearranging gives a differential equation for the general case of liquid pressure ψ,

$$g \frac{\partial K}{\partial z} + \frac{\partial K}{\partial z} \frac{\partial \psi}{\partial z} + K \frac{\partial^2 \psi}{\partial z^2} = \rho \frac{\mathrm{d}\theta}{\mathrm{d}\psi} \frac{\partial \psi}{\partial t}.$$

This is equivalent to

$$\frac{\partial}{\partial z} \left(K \frac{\partial \psi}{\partial z} \right) + g \frac{\partial K}{\partial z} = \rho \frac{\mathrm{d}\theta}{\mathrm{d}\psi} \frac{\partial \psi}{\partial t},$$

which is Richards' equation as originally presented by Richards' in [83], in the negative z direction. We use it in the form (5.3).

5.3 Properties of Richards' equation

We consider Richards' equation for the case of one-dimensional unsaturated flow (5.3)–(5.4) and demonstrate that the solution conserves mass (which is relevant when applying our moving mesh scheme), and has centre of mass which moves in one direction. These properties are proved in the same manner as in §4.3.

Lemma 5.3.1 *Richards' equation (5.3) conserves mass in time.*

Proof To prove that the total mass does not change over time we show that the derivative of the total mass (in time) is zero. Using the Leibnitz integral rule,

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} u(x,t)\,\mathrm{d}x = \int_{a(t)}^{b(t)} \frac{\partial u}{\partial t}\,\mathrm{d}x + u(b,t)\frac{\mathrm{d}b}{\mathrm{d}t} - u(a,t)\frac{\mathrm{d}a}{\mathrm{d}t}.$$

Substituting $\frac{\partial u}{\partial t}$ from (5.3), and noting that the last two terms vanish due to zero boundary conditions (5.4),

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} u(x,t)\,\mathrm{d}x = u(b,t)^{n-2}\frac{\partial b}{\partial x} + u(b,t)^n - u(a,t)^{n-2}\frac{\partial a}{\partial x} + u(a,t)^n.$$

The right-hand side is zero, again due to the boundary conditions (5.4), hence

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} u(x,t)\,\mathrm{d}x \quad = \quad 0,$$

as required. □

Lemma 5.3.2 *For Richards' equation (5.3) the centre of mass always moves in one direction.*

Proof The centre of mass $\bar{x}(t)$ is given by the ratio (4.9), as in §4.3. Since mass is conserved, we consider the numerator of (4.9) only. Differentiating the numerator with respect to time, using the Leibnitz integral rule, gives

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} x\,u(x,t)\,\mathrm{d}x = \int_{a(t)}^{b(t)} x\frac{\partial u}{\partial t}\,\mathrm{d}x + b(t)u(b,t)\frac{\partial b}{\partial x} - a(t)u(a,t)\frac{\partial a}{\partial x}.$$

Substituting $\frac{\partial u}{\partial t}$ from (5.3), and noting that the last two terms vanish due to zero boundary conditions (5.4),

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} x\,u(x,t)\,\mathrm{d}x = \int_{a(t)}^{b(t)} \left\{ x\frac{\partial}{\partial x}\left(u(x,t)^{n-2}\frac{\partial u}{\partial x} + u(x,t)^n \right) \right\}\,\mathrm{d}x.$$

Using integration by parts on the right-hand side gives

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} x\,u(x,t)\,\mathrm{d}x \quad &= \quad b(t)\left(u(b,t)^{n-2}\frac{\partial b}{\partial x} + u(b,t)^n \right) - a(t)\left(u(a,t)^{n-2}\frac{\partial a}{\partial x} + u(a,t)^n \right) \\ &\qquad - \int_{a(t)}^{b(t)} \left\{ u(x,t)^{n-2}\frac{\partial u}{\partial x} + u(x,t)^n \right\}\,\mathrm{d}x.\end{aligned}$$

The $u(a,t)$ and $u(b,t)$ terms vanish due to the boundary conditions (5.4). This leaves only the integral on the right-hand side, which can be rearranged such that

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} x\,u(x,t)\,\mathrm{d}x &= -\frac{1}{n-1}\int_{a(t)}^{b(t)}\frac{\partial}{\partial x}u(x,t)^{n-1}\,\mathrm{d}x-\int_{a(t)}^{b(t)}u(x,t)^n\,\mathrm{d}x,\\ &= -\frac{1}{n-1}\Big[u(b,t)^{n-1}-u(a,t)^{n-1}\Big]-\int_{a(t)}^{b(t)}u(x,t)^n\,\mathrm{d}x.\end{aligned}$$

Once more, the $u(a,t)$ and $u(b,t)$ terms vanish due to the boundary conditions (5.4), leaving

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{b(t)} x\,u(x,t)\,\mathrm{d}x = -\int_{a(t)}^{b(t)}u(x,t)^n\,\mathrm{d}x \quad < \quad 0,$$

which is strictly negative for $u > 0$, indicating that the centre of mass moves in the direction of the negative x-axis. □

By showing that the mass is conserved, we know that updating the total mass at each time-level is not required when applying our moving mesh method. In addition, knowing that the centre of mass moves in one direction allows us to check that our numerical solution is exhibiting expected behaviour. Before numerically solving Richards' equation, we seek a self-similar solution to Richards' equation in the next section.

5.4 A self-similar solution

In this section we describe a class of exact solutions to Richards' equation (5.3) that are invariant under a scaling group in the variables (t,x,u), and therefore take the so-called self-similar form. A definition of self-similarity, from [11], is given in §4.4 where we derived a self-similar solution for the PME. We use the same procedure from §4.4, therefore we begin by determining a scale-invariant transformation of Richards' equation (5.3). Although the same notation is used in this section as for the PME, the variables are in fact different.

5.4.1 Scale invariance

Consider the scaling transformation

$$t=\lambda\hat{t}, \qquad x=\lambda^{\beta}\hat{x}, \qquad u=\lambda^{\gamma}\hat{u}, \tag{5.8}$$

where λ is the scaling parameter and β and γ are constants.

Transforming the derivatives in the PDE into variables $\hat{t}$, $\hat{x}$ and $\hat{u}$ gives

$$\begin{aligned} \frac{\partial u}{\partial t} &= \lambda^{\gamma-1}\frac{\partial \hat{u}}{\partial \hat{t}}, && (5.9)\\ \frac{\partial u}{\partial x} &= \lambda^{\gamma-\beta}\frac{\partial \hat{u}}{\partial \hat{x}}, && (5.10) \end{aligned}$$

as with the PME. For Richards' equation we also have the terms

$$\begin{aligned} u(x,t)^{n-2} &= \lambda^{\gamma(n-2)}\hat{u}(\hat{x},\hat{t})^{n-2}, && (5.11)\\ u(x,t)^{n} &= \lambda^{n\gamma}\hat{u}(\hat{x},\hat{t})^{n}. && (5.12) \end{aligned}$$

The transformed left-hand side of (5.3) is (5.9). Using (5.10)–(5.12) we transform the right-hand side to obtain

$$\frac{\partial}{\partial x}\left(u(x,t)^{n-2}\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial x}(u(x,t)^n) \;=\; \lambda^{\gamma(n-1)-2\beta}\frac{\partial}{\partial \hat{x}}\left(\hat{u}(\hat{x},\hat{t})^{n-2}\frac{\partial \hat{u}}{\partial \hat{x}}\right)+\lambda^{\gamma n-\beta}\frac{\partial}{\partial \hat{x}}(\hat{u}(\hat{x},\hat{t})^n).$$

Hence, Richards' equation (5.3) transformed by (5.8) is

$$\lambda^{\gamma-1}\frac{\partial \hat{u}}{\partial \hat{t}}=\lambda^{\gamma(n-1)-2\beta}\frac{\partial}{\partial \hat{x}}\left(\hat{u}(\hat{x},\hat{t})^{n-2}\frac{\partial \hat{u}}{\partial \hat{x}}\right)+\lambda^{\gamma n-\beta}\frac{\partial}{\partial \hat{x}}(\hat{u}(\hat{x},\hat{t})^n).$$

Therefore, for the PDE (5.3) to be invariant under the transform (5.8) we require

$$\gamma-1=\gamma(n-1)-2\beta=n\gamma-\beta. \tag{5.13}$$

We consider (5.13) as two equalities in order to determine β and γ,

$$\gamma-1=\gamma(n-1)-2\beta \quad \text{and} \quad \gamma(n-1)-2\beta=n\gamma-\beta.$$

Thus

$$\gamma=-\beta=-\frac{1}{n}. \tag{5.14}$$

Note that $\gamma+\beta=0$, which is consistent with the total mass being scale invariant. From (5.14), the scale invariant transformation (5.8) becomes

$$t=\lambda\hat{t}, \qquad x=\lambda^{\frac{1}{n}}\hat{x}, \qquad u=\lambda^{-\frac{1}{n}}\hat{u}. \tag{5.15}$$

To summarise, the variables u, x and t can be rescaled as in (5.15) whilst still satisfying the PDE (5.3).

We now define scale-invariant similarity variables. From (5.8) we have

$$\lambda = \frac{t}{\hat{t}} = \frac{x^{\frac{1}{\beta}}}{\hat{x}^{\frac{1}{\beta}}} = \frac{u^{\frac{1}{\gamma}}}{\hat{u}^{\frac{1}{\gamma}}}.$$

Bearing this scaling in mind, we introduce the two new variables with β and γ as in (5.14)

$$\zeta = \frac{u}{t^{\gamma}} = \frac{\hat{u}}{\hat{t}^{\gamma}}, \tag{5.16}$$

$$\xi = \frac{x}{t^{\beta}} = \frac{\hat{x}}{\hat{t}^{\beta}}, \tag{5.17}$$

which are independent of λ and hence scale invariant under the transformation (5.8). We use (5.16) and (5.17) to find a self-similar solution for Richards' equation (5.3) in the next section.

5.4.2 Self-similarity

We obtain a self-similar solution of Richards' equation in the same manner as for the PME in §4.4.2, by assuming that there is a functional relationship $\zeta = \zeta(\xi)$ between the similarity variables (5.16) and (5.17), based on our rescaling (5.8). The self-similar solution is dependent only on the solution to an ODE that is obtained by transforming Richards' equation (5.3) into the variables ζ and ξ.

The transformed left-hand side of Richards' equation (5.3) is given by (4.19)

$$\frac{\partial u}{\partial t} = -\beta t^{\gamma-1}\xi\frac{\mathrm{d}\zeta}{\mathrm{d}\xi} + \zeta(\xi)\gamma t^{\gamma-1}. \tag{5.18}$$

We transform the right-hand side of Richards' equation in the same way that (4.19) was derived, so we first substitute in (5.16),

$$\frac{\partial}{\partial x}\left(u(x,t)^{n-2}\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial x}u(x,t)^{n} = \frac{\partial \xi}{\partial x}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\zeta(\xi)^{n-2}t^{\gamma(n-2)}\frac{\partial \xi}{\partial x}\frac{\mathrm{d}}{\mathrm{d}\xi}(\zeta(\xi)t^{\gamma})\right) + \frac{\partial \xi}{\partial x}\frac{\mathrm{d}}{\mathrm{d}\xi}(\zeta(\xi)^{n}t^{n\gamma}).$$

Substituting (5.17) into the above equation gives

$$\begin{aligned}\frac{\partial}{\partial x}\left(u(x,t)^{n-2}\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial x}u(x,t)^{n} &= t^{-\beta}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\zeta(\xi)^{n-2}t^{\gamma(n-2)}t^{\gamma-\beta}\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}\right) + t^{-\beta}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\zeta(\xi)^{n}t^{n\gamma}\right),\\ &= t^{\gamma(n-1)-2\beta}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\zeta(\xi)^{n-2}\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}\right) + t^{n\gamma-\beta}\frac{\mathrm{d}}{\mathrm{d}\xi}(\zeta(\xi)^{n}). \qquad (5.19)\end{aligned}$$

Putting together (5.18) and (5.19) gives an equation for ζ in terms of ξ,

$$-\beta t^{\gamma-1}\xi\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}+\zeta(\xi)\gamma t^{\gamma-1}=t^{(n-1)\gamma-2\beta}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\zeta(\xi)^{n-2}\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}\right)+t^{n\gamma-\beta}\frac{\mathrm{d}}{\mathrm{d}\xi}(\zeta(\xi)^{n}).$$

Note that t disappears from this equation since $2\gamma(n-1)-3\beta+1=0$ from (5.13). Substituting for γ and β from (5.14), gives the ODE

$$-\frac{1}{n}\left(\zeta(\xi)+\xi\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}\right) \quad = \quad \frac{\mathrm{d}}{\mathrm{d}\xi}\left(\zeta(\xi)^{n-2}\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}+\zeta(\xi)^{n}\right). \tag{5.20}$$

By moving all the terms to one side we have

$$\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\zeta(\xi)^{n-2}\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}+\zeta(\xi)^{n}\right)+\frac{1}{n}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\xi\zeta(\xi)\right)=0.$$

Integrating with respect to ξ gives

$$\zeta(\xi)^{n-2}\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}+\frac{\xi}{n}\zeta(\xi)+\zeta(\xi)^{n}=0. \tag{5.21}$$

where we have set the integration constant to zero since Richards' equation has zero boundary conditions, indicating that the transformed equation (5.20) has zero boundary conditions.

For the specific cases $n=2,3$ the ODE (5.21) is a Ricatti equation [50]. We now present a solution to the Ricatti equation for $n=3$.

The specific case $n = 3$

For $n=3$, equation (5.21) gives the Ricatti equation [50],

$$0=\zeta(\xi)\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}+\frac{\xi}{3}\zeta(\xi)+\zeta(\xi)^{3}.$$

To solve this equation we consider the transformation

$$\zeta(\xi)\equiv\frac{w_{\xi}}{w}, \tag{5.22}$$

which leads to the second order linear homogeneous equation

$$0=\frac{\partial^{2}w}{\partial\xi^{2}}+\frac{1}{3}\xi w.$$

Solutions to equations of this form are known to contain Bessel functions. The specific solution for positive ξ is

$$w(\xi) = \frac{1}{3}\sqrt{\xi}\left[AJ_{-\frac{1}{3}}\left(\frac{2}{3\sqrt{3}}\xi^{\frac{3}{2}}\right) + BJ_{\frac{1}{3}}\left(\frac{2}{3\sqrt{3}}\xi^{\frac{3}{2}}\right)\right], \tag{5.23}$$

where $J(\cdot)$ denotes a Bessel function of the first kind. We write (5.22) as

$$\zeta(\xi) = \frac{\mathrm{d}}{\mathrm{d}\xi}(\ln w),$$

and substitute in (5.23) to give the solution ζ as

$$\zeta = \frac{\mathrm{d}}{\mathrm{d}\xi}\ln\left\{\frac{1}{3}\sqrt{\xi}\left[AJ_{-\frac{1}{3}}\left(\frac{2}{3\sqrt{3}}\xi^{\frac{3}{2}}\right) + BJ_{\frac{1}{3}}\left(\frac{2}{3\sqrt{3}}\xi^{\frac{3}{2}}\right)\right]\right\}. \tag{5.24}$$

The final stage to find the self-similar solution for Richards' equation with $n = 3$ is to transform (5.24) using the results (5.16), (5.17), giving

$$u(x,t) = \frac{\mathrm{d}}{\mathrm{d}x}\ln\left\{\frac{1}{3}x^{\frac{1}{2}}t^{-\frac{1}{6}}\left[AJ_{-\frac{1}{3}}\left(\frac{2}{3\sqrt{3}}x^{\frac{3}{2}}t^{-\frac{1}{2}}\right) + BJ_{\frac{1}{3}}\left(\frac{2}{3\sqrt{3}}x^{\frac{3}{2}}t^{-\frac{1}{2}}\right)\right]\right\} \tag{5.25}$$

for positive x. Evaluating the differential and substituting $t = 1$ gives

$$u(x,1) = \frac{\sqrt{3}\left[AJ_{-\frac{1}{3}}(\mu) + BJ_{\frac{1}{3}}(\mu)\right] + 2x^{\frac{3}{2}}\left\{A\left[-J_{\frac{2}{3}}(\mu) - \frac{\sqrt{3}}{2x^{\frac{3}{2}}}J_{-\frac{1}{3}}(\mu)\right] + B\left[-J_{\frac{4}{3}}(\mu) + \frac{\sqrt{3}}{2x^{\frac{3}{2}}}J_{\frac{1}{3}}(\mu)\right]\right\}}{2\sqrt{3}x\left[AJ_{-\frac{1}{3}}(\mu) + BJ_{\frac{1}{3}}(\mu)\right]} \tag{5.26}$$

where $\mu = \left(\frac{2}{3\sqrt{3}}x^{\frac{3}{2}}\right)$ for convenience, and $x > 0$. A plot of (5.26) is given in Figure 5.1 for $A = 1$ and $B = 0.3621$. It is unclear that we are able to satisfy the zero boundary conditions at $a(t) = b(t) = 0$, so we do not pursue the self-similar solution any further. Therefore we compare our numerical results instead with those from a fixed mesh finite difference scheme on a very fine mesh.

In the next section we apply our moving mesh method to Richards' equation.

5.5 Moving meshes

Since mass is conserved in time we can use the moving mesh method described in §3.1, with the same notation, i.e. $\tilde{x}_j(t^m) \approx x_j^m$ denotes the jth node of the mesh with $N+1$ nodes, at time $m\Delta t$, $m = 0, 1\ldots$, and $u_j^m \approx \tilde{u}_j(t^m)$ and $v_j^m \approx \tilde{v}_j(t^m)$ denote the solution and mesh

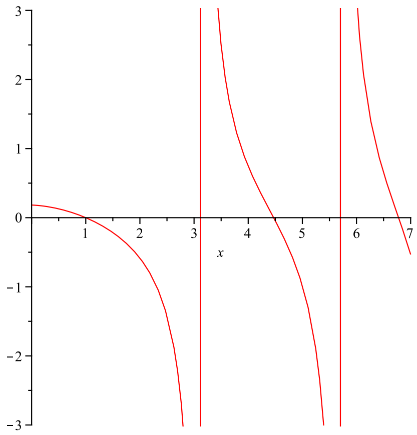


Fig. 5.1: A self-similar solution for Richards' equation.

velocity at these nodes.

We model the PDE (3.1) with

$$\mathcal{L}u \equiv \frac{\partial}{\partial x}\left(u(x,t)^{n-2}\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial x}u(x,t)^n, \tag{5.27}$$

imposing zero Dirichlet boundary conditions over the region $x(t) \in [a(t), b(t)]$.

We show that, given a mesh $\tilde{x}_j(t^m)$, with corresponding solution $\tilde{u}_j(t^m)$, we may calculate the updated mesh $\tilde{x}_j(t^{m+1})$ and solution $\tilde{u}_j(t^{m+1})$ by computing a mesh velocity $\tilde{v}_j(t^m)$.

5.5.1 Determining the mesh velocity

The mesh velocity is given by substituting (5.27) into (3.8),

$$\tilde{v}_j(t) = -\frac{1}{\tilde{u}_j(t)}\int_{a(t)}^{\tilde{x}(t)} \left\{\frac{\partial}{\partial x}\left(u(x,t)^{n-2}\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial x}u(x,t)^n\right\} \mathrm{d}x.$$

Hence, recalling the boundary condition $u(a,t) = 0$, the points move in time such that

$$\tilde{v}_j(t) = -\tilde{u}_j(t)^{n-3}\left.\frac{\partial u}{\partial x}\right|_{\tilde{x}_j(t)} - \tilde{u}_j(t)^{n-1},$$

where $j = 1, 2, \ldots, N-1$. This can also be written as

$$\tilde{v}_j(t) = -\frac{1}{n-2}\left.\frac{\partial(u^{n-2})}{\partial x}\right|_{\tilde{x}_j(t)} - \tilde{u}_j(t)^{n-1}. \tag{5.28}$$

When discretising (5.28) we consider two ways to approximate the derivative. The first gives

$$v_j^m = -\frac{1}{n-2}\left(\frac{(u^{n-2})_{j+1}^m - (u^{n-2})_{j-1}^m}{x_{j+1}^m - x_{j-1}^m}\right) - (u^{n-1})_j^m, \qquad j = 1, 2, ..., N-1, \tag{5.29}$$

which is a second order discretisation on a uniform mesh, but only a first order discretisation of (5.28) on an irregular mesh. The second approximation uses (4.37) to give a second order discretisation, such that

$$v_j^m = -\frac{1}{n-2}\left(\frac{\frac{1}{\Delta x_{j+}^m}\left(\frac{\Delta (u^{n-2})_{j+}^m}{\Delta x_{j+}^m}\right) + \frac{1}{\Delta x_{j-}^m}\left(\frac{\Delta (u^{n-2})_{j-}^m}{\Delta x_{j-}^m}\right)}{\frac{1}{\Delta x_{j+}^m} + \frac{1}{\Delta x_{j-}^m}}\right) - (u^{n-1})_j^m, \tag{5.30}$$

for $j = 1, \ldots, N-1$, where $\Delta(\cdot)_{j+} = (\cdot)_{j+1}^m - (\cdot)_j^m$ and $\Delta(\cdot)_{j-} = (\cdot)_j^m - (\cdot)_{j-1}^m$. The outer boundary velocities v_0^m, v_N^m are extrapolated by a polynomial approximation using (v_1^m, v_2^m, v_3^m) and $(v_{N-3}^m, v_{N-2}^m, v_{N-1}^m)$, or given by one-sided approximations of (5.29) or (5.30).

The new mesh x_j^{m+1} is obtained from v_j^m by a time-stepping scheme.

5.5.2 Recovering the solution

Once the updated mesh x_j^{m+1} has been determined, the updated solution u_j^{m+1}, $j = 1, \ldots, N-1$, is given by either (3.10) or (3.16), the latter being more accurate for a non-uniform mesh. At the boundaries $u_0^{m+1} = u_{N+1}^{m+1} = 0$ from (5.4).

5.5.3 The full algorithm

Given a mesh x_j^m, solution u_j^m, $j = 0, \ldots, N$, at $t = t^m$, $m \geq 0$:

- Compute the mesh velocity v_j^m from (5.29) or (5.30);
- Compute the updated mesh x_j^{m+1} by a time-stepping scheme;
- Compute the updated solution u_j^{m+1} from (3.10) or (3.16).

5.5.4 Time-stepping schemes

Explicit schemes

The simplest method to time-step the mesh is the first order explicit Euler time-stepping scheme,

$$\frac{x_j^{m+1} - x_j^m}{\Delta t} = v_j^m.$$

We substitute for v_j^m from (5.29) or (5.30). The explicit Euler time-stepping scheme requires small Δt so that the x_j^m remain stable, and to avoid mesh tangling. We also implemented the adaptive predictor-correcter Runge-Kutta methods in Matlab, and again found that the method does not lead to a stiff system.

A semi-implicit scheme

To determine a semi-implicit time-stepping scheme for solving Richards' equation we first consider the explicit time-stepping scheme using v_j^m from (5.29) with the spatial discretisation halved,

$$\frac{x_j^{m+1} - x_j^m}{\Delta t} = -\frac{1}{n-2}\frac{(u^{n-2})^m_{j+\frac{1}{2}} - (u^{n-2})^m_{j-\frac{1}{2}}}{x^m_{j+\frac{1}{2}} - x^m_{j-\frac{1}{2}}} - (u^{n-1})^m_j,$$

for $j = 1, 2, ..., N-1$. Assuming that the mesh $\tilde{x}_j(t)$ changes smoothly in time, we alter the Euler scheme to be semi-implicit in the manner

$$\frac{x_j^{m+1} - x_j^m}{\Delta t} = -\frac{1}{(n-2)\Delta x_j^m}\left((u^{n-2})^m_{j+\frac{1}{2}}\frac{\Delta x^{m+1}_{j-}}{\Delta x^m_{j-}} - (u^{n-2})^m_{j-\frac{1}{2}}\frac{\Delta x^{m+1}_{j+}}{\Delta x^m_{j+}}\right) - (u^{n-1})^m_j, \quad (5.31)$$

where $\Delta x_j^m = (x^m_{j+\frac{1}{2}} - x^m_{j-\frac{1}{2}})$, $\Delta x^m_{j-} = (x^m_j - x^m_{j-1})$ and $\Delta x^m_{j+} = (x^m_{j+1} - x^m_j)$. This semi-implicit scheme (5.31) is first order in time. Before calculating the internal nodes semi-implicitly by (5.31), the boundary nodes x_0^{m+1}, x_N^{m+1} are calculated by an explicit scheme enabling $\Delta x^{m+1}_{1_-} = (x_1^{m+1} - x_0^{m+1})$ and $\Delta x^{m+1}_{N-1_+} = (x_N^{m+1} - x^{m+1}_{N-1})$ to be determined.

Rearranging (5.31), and expanding the $\Delta x^{m+1}_{j\pm}$ terms gives

$$\begin{aligned}\frac{x_j^{m+1} - x_j^m}{\Delta t} = & -\frac{1}{(n-2)\Delta x_j^m \Delta x^m_{j+}\Delta x^m_{j-}}\Big[(u^{n-2})^m_{j+\frac{1}{2}}\Delta x^m_{j+}(x_j^{m+1} - x^{m+1}_{j-1}) \\ & - (u^{n-2})^m_{j-\frac{1}{2}}\Delta x^m_{j-}(x^{m+1}_{j+1} - x^{m+1}_j)\Big] - (u^{n-1})^m_j. \quad (5.32)\end{aligned}$$

Our moving mesh method moves the nodes such that partial masses of the solution are conserved, see equation (3.6) in §3.1. Bearing this in mind we define initial masses, which remain unchanged in time, as

$$\begin{aligned} D_{j_+} &= (u^{n-2})^0_{j+\frac{1}{2}} \Delta x^0_{j_+} = (u^{n-2})^m_{j+\frac{1}{2}} \Delta x^m_{j_+}, & (5.33) \\ D_{j_-} &= (u^{n-2})^0_{j-\frac{1}{2}} \Delta x^0_{j_-} = (u^{n-2})^m_{j-\frac{1}{2}} \Delta x^m_{j_-}, & (5.34) \end{aligned}$$

thus simplifying equation (5.32) to

$$\frac{x_j^{m+1} - x_j^m}{\Delta t} = -\frac{D_{j_+}(x_j^{m+1} - x_{j-1}^{m+1}) - D_{j_-}(x_{j+1}^{m+1} - x_j^{m+1})}{(n-2)\Delta x_j^m \Delta x_{j_+}^m \Delta x_{j_-}^m} - (u^{n-1})_j^m.$$

To determine the new mesh by the semi-implicit scheme we solve

$$A\mathbf{x}^{m+1} = \mathbf{b}^m,$$

where

$$\begin{aligned} \mathbf{x}^{m+1} &= \left[x_1^{m+1}, \cdots, x_{N-1}^{m+1}\right]^T, \\ \mathbf{b}^m &= \left[x_1^m, \cdots, x_{N-1}^m\right]^T - \Delta t\left[(u^{n-1})_1^m, \cdots, (u^{n-1})_{N-1}^m\right]^T, \end{aligned}$$

and A is a tridiagonal matrix with lower, main and upper diagonals Al_j, Ad_j and Au_j,

$$\begin{aligned} Al_j &= -\frac{c_{j_+}\Delta t}{(n-2)\Delta x_j^m \Delta x_{j_+}^m \Delta x_{j_-}^m}, & (5.35) \\ Au_j &= -\frac{c_{j_-}\Delta t}{(n-2)\Delta x_j^m \Delta x_{j_+}^m \Delta x_{j_-}^m}, & (5.36) \\ Ad_j &= 1 - Al_j - Au_j. & (5.37) \end{aligned}$$

The right-hand side of the semi-implicit scheme (5.31) has the additional term $(u^{n-1})_j^m$, so it is not in the form (3.39). Thus Theorem 3.4.1 does not hold, and therefore we have not ensured that mesh tangling will not occur. Hence, we consider an alternative moving mesh method such that the semi-implicit scheme is of the form (3.39), thus Theorem 3.4.1 will hold. We consider this alternative approach in the next section.

5.6 An alternative moving mesh method

The moving mesh method described in §3.1 utilises

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{\tilde{x}_j(t)} u(x,t)\,\mathrm{d}x = 0,$$

a relation arising from the conservation of mass property. When applied to Richards' equation, this gives a mesh velocity (5.28). An alternative approach is to balance the rate of increase of mass in a subregion by the flux term. This is equivalent to balancing the right-hand side of (3.7) with the flux term, giving

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{\tilde{x}_j(t)} u(x,t)\,\mathrm{d}x = \int_{a(t)}^{\tilde{x}_j(t)} \frac{\partial}{\partial x}u(x,t)^n\,\mathrm{d}x = \tilde{u}_j(t)^n, \tag{5.38}$$

since $u = 0$ at $a(t)$. To determine a mesh velocity from (5.38) we define the partial masses at time t and interval points j. This is a similar approach to that in §3.3 where the rate of increase of mass is balanced by the *source* term. Subsequently, the procedure is similar so we first define the partial masses to be

$$\Theta_j(t) = \int_{a(t)}^{\tilde{x}_j(t)} u(x,t)\,\mathrm{d}x, \quad j = 0,\ldots,N. \tag{5.39}$$

Given a mesh $\tilde{x}_j(t)$ and solution $\tilde{u}_j(t)$, we can evaluate the partial masses $\Theta_j(t)$ directly from (5.39). To evaluate updated values of the partial mass (which are required for determining the updated solution) we compute $\dot{\Theta}_j(t)$, and then approximate the updated total mass using a semi-implicit time-stepping scheme. Simultaneously, the mesh velocity $\tilde{v}_j(t)$ is computed, such that the mesh and partial masses are updated together. This ultimately enables us to recover the updated solution on the new mesh. Details are given in the following subsections.

5.6.1 Determining the rate of change partial masses

To evaluate the updated partial masses (which are required for determining the updated solution) we compute $\dot{\Theta}_j(t)$ from (5.38) as

$$\dot{\Theta}_j(t) = \tilde{u}_j(t)^n. \tag{5.40}$$

The discrete form of (5.40) is simply

$$\dot{\Theta}_j^m = (u^n)_j^m. \tag{5.41}$$

From (5.41) we can determine $\Theta_j(t^{m+1})$ by, say, the explicit Euler scheme

$$\Theta_j^{m+1} = \Theta_j^m + \Delta t \dot{\Theta}_j^m.$$

Note that $\Theta_N^0 = \Theta_N^m$ is known for all $m = 0, 1, 2, \ldots$ because the total mass remains constant.

5.6.2 Determining the mesh velocity

We now calculate the new mesh velocity $\tilde{v}_j(t)$ by differentiating (5.39) with respect to time using the Leibnitz integral rule to give

$$\dot{\Theta}_j(t) = \frac{\mathrm{d}}{\mathrm{d}t}\int_{a(t)}^{\tilde{x}_j(t)} u(x,t)\,\mathrm{d}x \quad = \quad \int_{a(t)}^{\tilde{x}_j(t)} \frac{\partial u}{\partial t}\,\mathrm{d}x + \tilde{u}_j(t)\tilde{v}_j(t) - u(a,t)v(a,t).$$

Substituting $\frac{\partial u}{\partial t}$ from (5.3), evaluating the integral, and using the boundary condition $u(a,t) = 0$ from (5.4),

$$\dot{\Theta}_j(t) \quad = \quad \tilde{u}_j(t)^{n-2}\left.\frac{\partial u}{\partial x}\right|_{\tilde{x}_j(t)} + \tilde{u}_j(t)^n + \tilde{u}_j(t)\tilde{v}_j(t). \tag{5.42}$$

Substituting (5.42) for the left-hand side of (5.40) (which is from the relation (5.38)),

$$\tilde{u}_j(t)^{n-2}\left.\frac{\partial u}{\partial x}\right|_{\tilde{x}_j(t)} + \tilde{u}_j(t)^n + \tilde{u}_j(t)\tilde{v}_j(t) = \tilde{u}_j(t)^n.$$

Thus, an alternative mesh velocity for Richards' equation is

$$\begin{aligned} \tilde{v}_j(t) \quad &= \quad -\tilde{u}_j(t)^{n-3}\left.\frac{\partial u}{\partial x}\right|_{\tilde{x}_j(t)}, \\ &= \quad -\frac{1}{n-2}\left.\frac{\partial(u^{n-2})}{\partial x}\right|_{\tilde{x}_j(t)}, \end{aligned} \tag{5.43}$$

which is the mesh velocity equation (5.28) without the last term. To approximate (5.43) we could use a central difference approximation

$$v_j^m = -\frac{1}{n-2}\left(\frac{(u^{n-2})_{j+1}^m - (u^{n-2})_{j-1}^m}{x_{j+1}^m - x_{j-1}^m}\right), \tag{5.44}$$

which is a second order discretisation on a uniform mesh, but only a first order discretisation of (5.43) on an irregular mesh. An alternative approximation uses (4.37) to give a second order discretisation, such that

$$v_j^m = -\frac{1}{n-2}\left(\frac{\frac{1}{\Delta x_{j_+}^m}\left(\frac{\Delta(u^{n-2})_{j_+}^m}{\Delta x_{j_+}^m}\right)+\frac{1}{\Delta x_{j_-}^m}\left(\frac{\Delta(u^{n-2})_{j_-}^m}{\Delta x_{j_-}^m}\right)}{\frac{1}{\Delta x_{j_+}^m}+\frac{1}{\Delta x_{j_-}^m}}\right), \tag{5.45}$$

for $j = 1, \ldots, N-1$, where $\Delta(\cdot)_{j_+} = (\cdot)_{j+1}^m - (\cdot)_j^m$ and $\Delta(\cdot)_{j_-} = (\cdot)_j^m - (\cdot)_{j-1}^m$. The outer boundary velocities v_0^m, v_N^m are extrapolated by a polynomial approximation using (v_1^m, v_2^m, v_3^m) and $(v_{N-3}^m, v_{N-2}^m, v_{N-1}^m)$, or given by one-sided approximations of (5.29) or (5.30).

The new mesh x_j^{m+1} is obtained from v_j^m by a time-stepping scheme.

5.6.3 Recovering the solution

The approximations to the partial masses Θ_j^{m+1} and the mesh x_j^{m+1} are updated from $\dot{\Theta}_j^m$ and v_j^m, respectively, using a semi-implicit time-stepping scheme. Then the solution is updated using the approach of §3.3, namely, by either (3.34) or (3.35), the latter being more accurate for a non-uniform mesh. At the outer boundary, $u_{N+1}^{m+1} = 0$ from (4.31).

5.6.4 The full algorithm

Given a mesh x_j^m, solution u_j^m, $j = 0, \ldots, N$, at $t = t^m$, $m \geq 0$:

- Compute the mesh velocity v_j^m from (5.44) or (5.45);
- Compute the updated mesh x_j^{m+1} by a semi-implicit time-stepping scheme;
- Compute the updated solution u_j^{m+1} from (3.34) or (3.35).

5.6.5 A semi-implicit time-stepping scheme

A semi-implicit approach to update the mesh velocity for the alternative approach, assuming that the mesh moves smoothly with time, is

$$v_j^m = -\frac{1}{(n-2)\Delta x_j^m}\left((u^{n-2})_{j+\frac{1}{2}}^m \frac{\Delta x_{j_-}^{m+1}}{\Delta x_{j_-}^m} - (u^{n-2})_{j-\frac{1}{2}}^m \frac{\Delta x_{j_+}^{m+1}}{\Delta x_{j_+}^m}\right), \tag{5.46}$$

where $\Delta x_j^m = (x_{j+\frac{1}{2}}^m - x_{j-\frac{1}{2}}^m)$, $\Delta x_{j_-}^m = (x_j^m - x_{j-1}^m)$ and $\Delta x_{j_+}^m = (x_{j+1}^m - x_j^m)$ as before. This semi-implicit scheme (5.46) is first order in time. Before calculating the internal nodes semi-implicitly by (5.46), the boundary nodes x_0^{m+1}, x_N^{m+1} are calculated by an explicit scheme enabling $\Delta x_{1_-}^{m+1} = x_1^{m+1} - x_0^{m+1}$ and $\Delta x_{N-1_+}^{m+1} = x_N^{m+1} - x_{N-1}^{m+1}$ to be determined.

Rearranging (5.46), and expanding the $\Delta x_{j\pm}^{m+1}$ terms gives

$$\begin{aligned}\frac{x_j^{m+1} - x_j^m}{\Delta t} &= - \frac{1}{(n-2)(\Delta x)_j^m \Delta x_{j_+}^m \Delta x_{j_-}^m} \Big((u^{n-2})_{j+\frac{1}{2}}^m \Delta x_{j_+}^m (x_j^{m+1} - x_{j-1}^{m+1}) \\ &- (u^{n-2})_{j-\frac{1}{2}}^m \Delta x_{j_-}^m (x_{j+1}^{m+1} - x_j^{m+1}) \Big). \end{aligned} \tag{5.47}$$

Equation (5.47) is nearly identical to our earlier semi-implicit scheme for Richards' equation (5.32), the difference being that (5.47) does not have the last term present in (5.32). Subsequently, the updated mesh x_j^{m+1} is derived by solving the matrix system

$$A\mathbf{x}^{m+1} = \mathbf{x}^m,$$

where $\mathbf{x}^{m+1} = (x_1^{m+1}, \cdots x_{N-1}^{m+1})^T$, $\mathbf{x}^m = (x_1^m, \cdots x_{N-1}^m)^T$, and A is a tridiagonal matrix defined as before (with diagonals given by (5.35)–(5.37)).

Remark 5.6.1 *As in Remark 4.5.2, the implicitness of the semi-implicit approach, together with the explicit end point calculation, can be improved by using it in a predictor-corrector mode: solving the matrix system repeatedly within one time-level, whilst updating the explicit end point value x_N^{m+1}. The iterations affect the entries in the matrix A and vector $\mathbf{x}^{m+1}$, but the vector $\mathbf{x}^m$ remain unchanged at each time-level. We use the notation $(\cdot)^p$ to denote the iterations at each time-level. After one iteration ($p = 1$) the entries of A are at the new time-level $(m+1)$, giving $(x_j^{m+1})^1$ terms, and the entries of $\mathbf{x}^{m+1}$ are $(x_j^{m+1})^2$. This iteration process at each time-level modifies (5.47) to become*

$$\frac{(x_j^{m+1})^{p+1} - x_j^m}{\Delta t} = - \frac{D_{j_+}(x_j^{m+1} - x_{j-1}^{m+1})^{p+1} - D_{j_-}(x_{j+1}^{m+1} - x_j^{m+1})^{p+1}}{(n-2)(\Delta x_j^{m+1})^p (\Delta x_{j_+}^{m+1})^p (\Delta x_{j_-}^{m+1})^p}, \tag{5.48}$$

where $p = 0, 1, 2, \ldots$ and $D_{j\pm}$ are given by (5.33) and (5.34). If (5.48) is convergent it leads to the fully implicit scheme

$$\frac{x_j^{m+1} - x_j^m}{\Delta t} = - \frac{1}{(n-2)\Delta x_j^{m+1}} \left(\frac{D_{j_+}}{\Delta x_{j_+}^{m+1}} - \frac{D_{j_-}}{\Delta x_{j_-}^{m+1}} \right).$$

The semi-implicit scheme is now of the form described in §3.4.2 (but unlike the semi-implicit scheme (5.32)), we can show that the mesh does not tangle when using (5.47) by proving a maximum principle.

Ensuring monotonicity The maximum principle states that the maximum of x occurs at the boundary. To prove that this is the case for the semi-implicit scheme (5.46), we use exactly the same approach used in §3.4.2, with the same outcome: for each $j = 1, 2, ..., N-1$, x_j^m is bounded by its neighbours. Hence, the mesh is monotonic in space and is bounded by its endpoint values, so that overlapping cannot occur.

In the last two sections we have given the details for applying the moving mesh method to Richards' equation. In the next section we present the numerical results.

5.7 Numerical results

In this section we present results from applying the moving mesh method of §3.1 to Richards' equation as described in §5.5, and the alternative method described in §5.6. To test that the numerical solution from the moving mesh method converges (with the explicit Euler and semi-implicit time-stepping schemes), we compare the solution with that from a very fine fixed mesh. We also show convergence of the method in §5.6 using the semi-implicit time-stepping scheme.

All numerical results presented here take $n = 3$ and start with a equispaced mesh and use a second order approximation (equation (5.30) for the method in §5.5 and equation (5.45) for the method in §5.6) to calculate the mesh velocity v_j^m since they use more information to obtain the derivative at a node. However, for the method in §5.5 we also examined the numerical solution using (5.29) and it was noted that in the tests they gave the same results to at least $\mathcal{O}(10^{-3})$ compared with using (5.30). Similarly, we used (3.16) to recover the solution for the method in §5.5, since it is more accurate for a non-uniform mesh. All the same, we examined the numerical solution using the first order approximation (3.10) and found that in the tests, the numerical solutions were the same to at least $\mathcal{O}(10^{-2})$. The relatively small differences in the numerical solutions from using these different approaches suggests that the mesh remains fairly uniform when used to numerically solve Richards' equation. However, the same comparisons from the PME results gave much smaller differences, $\mathcal{O}(10^{-11})$ and $\mathcal{O}(10^{-12})$. Hence, we conclude that the mesh for the PME is more uniformly spread than for Richards' equation. This is probably since the PME has a symmetrical solution, whilst Richards' equation has a non-symmetrical solution. Physically, the skewness which is apparent in the solution to Richards' equation relates to the gravitational pull on the liquid in the downward direction (which is shown here as being along the x axis).

We look at the convergence of:

- the moving mesh method given in §5.5 with explicit Euler time-stepping;
- the moving mesh method given in §5.5 with semi-implicit time-stepping;

- the moving mesh method given in §5.6 with semi-implicit time-stepping;

as the number of nodes N increases and Δt decreases. We solve for $t \in [0, 0.5]$ and compute results for $N = 10 \times 2^{\hat{N}-1}$, $\hat{N} = 1, \ldots, 5$. We use the same notation given in §4.7 for the PME to compute both $x_{2^{\hat{N}-1}i,\hat{N}}$ and $u_{2^{\hat{N}-1}i,\hat{N}} \approx u(x_{2^{\hat{N}-1}i,\hat{N}}, 0.5)$ for each $i = 0, \ldots, 10$ as $\hat{N}$ increases. We compare these numerical solutions with the numerical solution calculated by solving Richards' equation on the fixed mesh $\bar{x}_{\bar{j}} \in [-4, 4]$, $\bar{j} = 0, 1, \ldots, 10000$, which is given by

$$\frac{\bar{u}^{m+1}_{\bar{j}+\frac{1}{2}} - \bar{u}^{m}_{\bar{j}+\frac{1}{2}}}{\Delta t} = (\bar{u}^{n-2})^{m}_{\bar{j}+\frac{1}{2}} \frac{\bar{u}^{m}_{\bar{j}+1} - \bar{u}^{m}_{\bar{j}}}{h} + (\bar{u}^{n})^{m}_{\bar{j}+\frac{1}{2}} - (\bar{u}^{n-2})^{m}_{\bar{j}-\frac{1}{2}} \frac{\bar{u}^{m}_{\bar{j}} - \bar{u}^{m}_{\bar{j}-1}}{h} - (\bar{u}^{n})^{m}_{\bar{j}-\frac{1}{2}},$$

where $h = 8 \times 10^{-4}$ is the uniform spacing between two mesh points, $(\bar{u}^n)^m_{\bar{j}+\frac{1}{2}} \approx \frac{1}{2}\big((\bar{u}^n)^m_{\bar{j}+1} + (\bar{u}^n)^m_{\bar{j}}\big)$ and $(\bar{u}^n)^m_{\bar{j}-\frac{1}{2}} \approx \frac{1}{2}\big((\bar{u}^n)^m_{\bar{j}} + (\bar{u}^n)^m_{\bar{j}-1}\big)$. We consider the $n = 3$ case and use the initial conditions

$$u(x, 0) = 1 - x^2, \qquad x \in [-1, 1],$$

with zero boundary conditions (5.3). To balance the spatial and temporal errors, we use $\Delta t = \mathcal{O}(\frac{1}{N^2})$, precisely $\Delta t = \frac{2}{5(4^{\hat{N}})}$ (as with the PME). We use $x_{2^{\hat{N}-1}i,\hat{N}}$ and find the closest match along the fixed mesh $\bar{x}_{\bar{j}}$ (where the fixed mesh points are different to 4 decimal places), then the corresponding solution on the fixed mesh $\bar{u}^m_{\bar{j}}$ is compared to $u_{2^{\hat{N}-1}i,\hat{N}}$.

As a measure of the errors, we calculate

$$E_N(u) = \sqrt{\frac{\sum_{i=0}^{10}(\bar{u}_{\bar{j}} - u_{2^{\hat{N}-1}i,\hat{N}})^2}{\sum_{i=0}^{10}(\bar{u}_{\bar{j}})}},$$

for $\hat{N} = 1, \ldots, 5$ (i.e. $N = 10, 20, 40, 80, 160$). We investigate the same hypothesis from our work on the numerical solution of the PME: that

$$E_N(u) \sim \frac{1}{N^p}, \tag{5.49}$$

holds for large N, where p is the estimated order of convergence. If (5.49) holds then we would expect that p_{2N} defined by

$$p_{2N} = -\log_2\left(\frac{E_{2N}(u)}{E_N(u)}\right),$$

would approach the constant values p as $N \to \infty$. Since each step of our scheme is second order in space and first order in time, and recalling that $\Delta t = \mathcal{O}\left(\frac{1}{N^2}\right)$, we might expect to see $p \approx 2$.

	§5.5, explicit		§5.5, semi-implicit		§5.6, semi-implicit	
N	$E_N(u)$	p_N	$E_N(u)$	p_N	$E_N(u)$	p_N
10	4.47×10^{-2}	-	9.78×10^{-2}	-	8.70×10^{-2}	-
20	7.97×10^{-3}	2.5	1.54×10^{-2}	2.7	4.33×10^{-2}	1.0
40	1.90×10^{-3}	2.1	3.69×10^{-3}	2.1	2.21×10^{-2}	1.0
80	4.75×10^{-4}	2.0	9.25×10^{-4}	2.0	1.14×10^{-2}	1.0
160	3.45×10^{-4}	0.5	3.63×10^{-4}	1.4	5.60×10^{-2}	1.0

Table 5.1: Relative errors for u with rates of convergence.

Convergence results are shown in Table 5.1. We discovered that the moving mesh value $x_{2^{\hat{N}-1}i,\hat{N}}$ rarely corresponded to the closest fixed mesh value $\bar{x}_{\bar{j}}$ by more than 3 decimal places. As a result, investigating larger N caused difficulty since $x_{2^3 i,4}$ and $x_{2^4 i,5}$ are frequently the same to 3 decimal places. This problem may account for the anomalies for p_{160}. All the same, we see that generally $E_N(u)$ decreases as N increases for each of the moving mesh methods. This strongly suggests that as the number of nodes increases, the solution $\tilde{u}_j(t)$ is converging in all three cases. The p-values presented imply second-order convergence of the solution $\tilde{u}_j(t)$ for the method in §5.5 and first-order convergence for the alternative method in §5.6.

Having established convergence of our moving mesh schemes we examine the numerical results. We observe from Figure 5.2 that the moving mesh method, used with an explicit Euler time-stepping scheme, successfully solves Richards' equation numerically. We also note from Figure 5.2(b) that the mesh moves smoothly and does not tangle.

Using a semi-implicit time-stepping scheme gives very similar results when compared to the results obtained when using the explicit Euler time-stepping scheme. However, we can take larger time-steps when using the semi-implicit scheme. Figure 5.3(a) shows that the nodes move in a very similar manner for the explicit Euler time-stepping scheme, with $\Delta t = 0.01$, and the semi-implicit time-stepping scheme, with $\Delta t = 0.02$. However, from Figure 5.3(b) we observe that the difference between the two approaches increases when the semi-implicit scheme takes a larger time-step of $\Delta t = 0.1$, which has resulted in a greater discrepancy of the boundary position. This indicates that although we can take larger time-steps with the semi-implicit scheme and it appears that the mesh does not tangle, we must still take modest time-steps to ensure accuracy. This is demonstrated further in Figure 5.4 which shows the solution from taking two very large time-steps of $\Delta t = 1$. We notice that the mesh does not tangle, but the solution at $t = 2$ is not very similar to the solution at $t = 2$ in Figure 5.2(a).

The semi-implicit time-stepping approach (4.45) is not in the form of (3.39), so Theorem 3.4.1 does not hold, as mentioned at the end of §5.5. Nonetheless, we observe from

Figures 5.3(a) and 5.3(b) that the mesh does not tangle for large time-steps. Using the alternative moving mesh method in §5.6 we ensured monotocity with a semi-implicit time-stepping scheme. We observe that this method also successfully solves Richards' equation numerically, see Figure 5.5(a). However, we notice from Figure 5.5(b) that the left boundary moves out more slowly, suggesting that the solution from the mass conserving method and the alternative method differ slightly. Since we have established a reasonable level of accuracy using the mass conserving method (see Table 5.1), we conclude that the difference between the two methods implies that the alternative method is less accurate. Inaccuracies may have occurred since the partial masses $\Theta_j(t)$ and mesh $\tilde{x}_j(t)$ are not updated simultaneously; the partial masses are updated explicitly, whilst the mesh is updated semi-implicitly. We also note from Figure 5.5(b) that the nodes seem drawn to the right, which is different from the use of the moving mesh scheme in §5.5, where the nodes are fairly evenly spread. The comparison is clearer in Figure 5.5(c), where we observe that the nodes are not spread as uniformly compared with the alternative method.

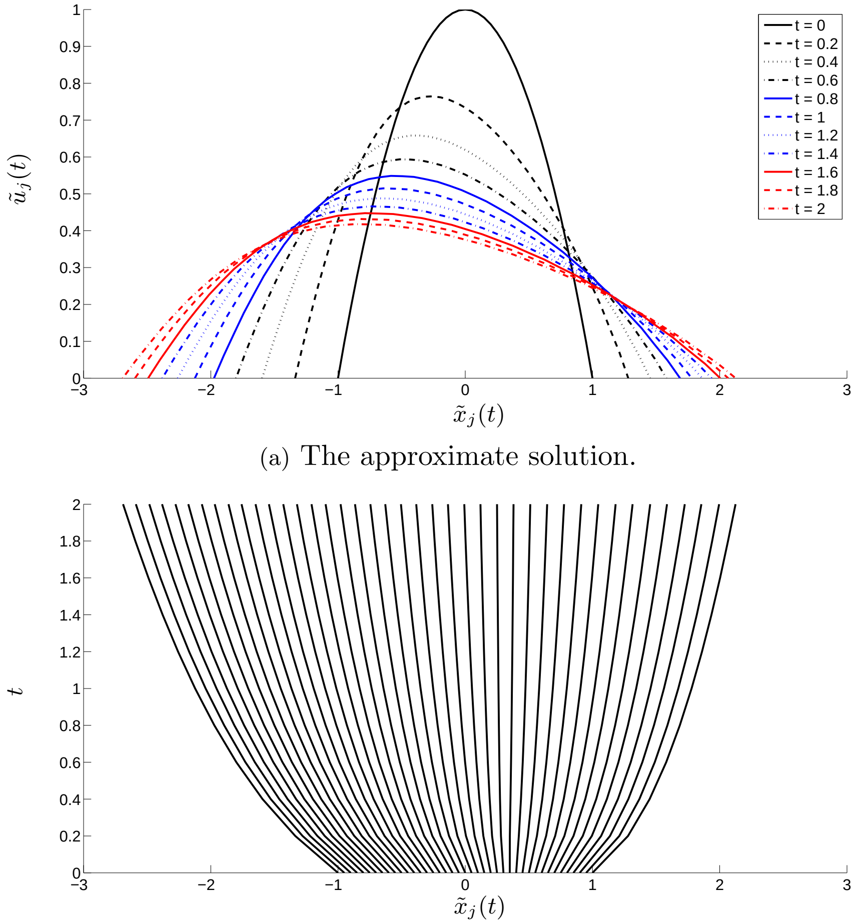
We have completed our application of our finite difference moving mesh method applied to Richards' equation. In the next section we apply the finite element moving mesh method of Baines, Hubbard and Jimack [5] to Richards' equation.

5.8 Using finite elements

We use the finite element moving mesh method in §2.3 to solve Richards' equation (5.3) with zero Dirichlet boundary conditions (5.4). Since Baines, Hubbard and Jimack [5] do not numerically solve Richards' equation, we provide details about the application of the finite element approach.

Referring to the algorithm given in §2.3.3 we note that we need an expression for the mesh velocity $v(x,t)$ and the solution $u(x,t)$. We do not require an expression for the rate of change of total mass since the mass remains constant (as with the PME). To determine the mesh velocity, we first determine an expression for the velocity potential of the mesh ψ. The velocity potential ψ is given by substituting (5.27) into the one-dimensional form of (2.19),

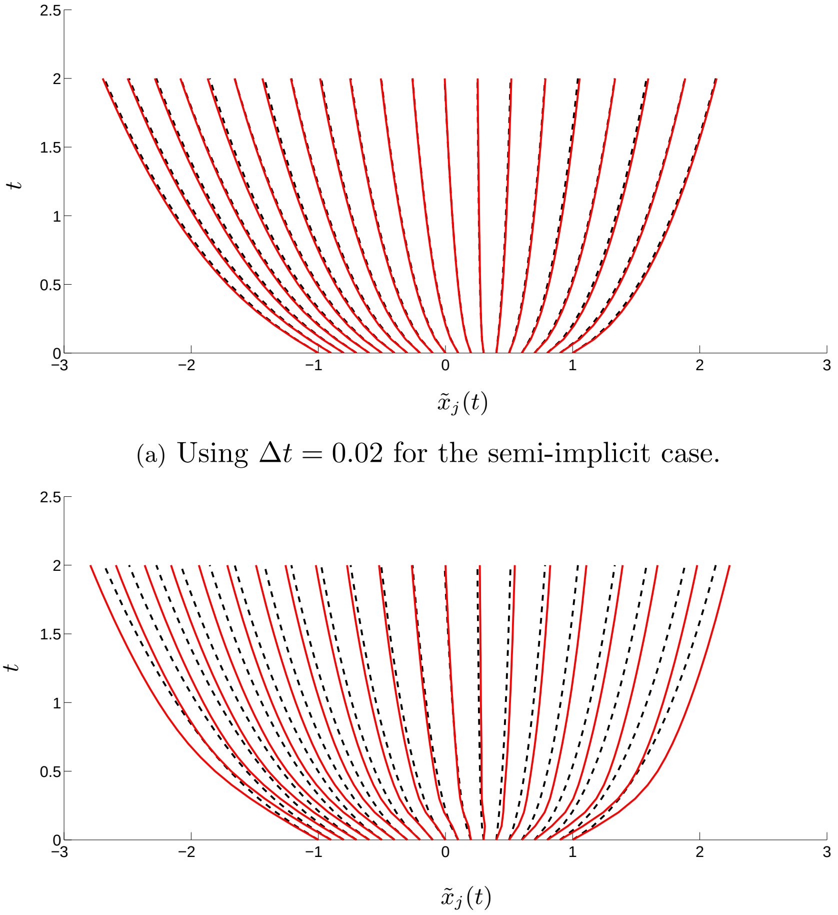
$$-\left[w_i u \frac{\partial \psi}{\partial x}\right]_{a(t)}^{b(t)} + \int_{a(t)}^{b(t)} u \frac{\partial \psi}{\partial x} \frac{\partial w_i}{\partial x}\, \mathrm{d}x = \int_{a(t)}^{b(t)} w_i \left[\frac{\partial}{\partial x}\left(u^{n-2}\frac{\partial u}{\partial x} + u^n\right)\right] \mathrm{d}x.$$



(a) The approximate solution.

(b) The mesh trajectory.

Fig. 5.2: Richards' equation with $n = 3$, $N = 40$, $\Delta t = 0.01$ and explicit time-stepping.



(a) Using $\Delta t = 0.02$ for the semi-implicit case.

(b) Using $\Delta t = 0.1$ for the semi-implicit case.

Fig. 5.3: The mesh movement using the explicit Euler time-stepping scheme with $\Delta t = 0.01$ (dashed black) and the semi-implicit time-stepping with Δt specified for each plot (solid red), $n = 3$, $N = 20$.

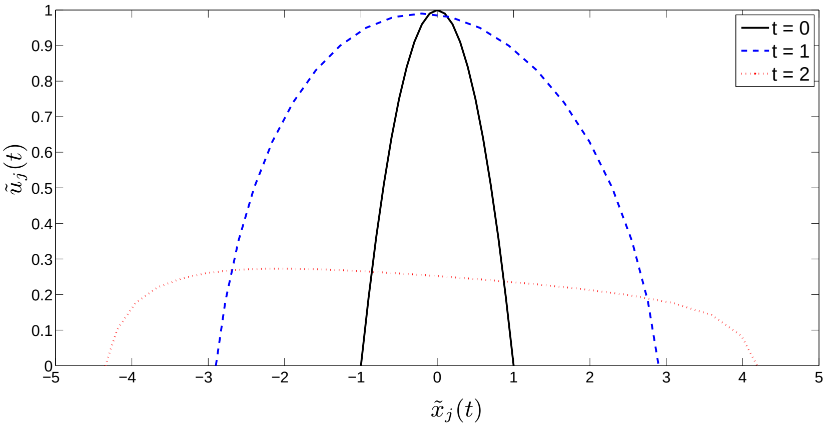


Fig. 5.4: Richards' equation with $n = 3$, $N = 20$, $\Delta t = 1$ and semi-implicit time-stepping.

We apply integration by parts to the right-hand side, and eliminate the first term of the left-hand side since we have zero boundary conditions, which gives

$$\int_{a(t)}^{b(t)} u\frac{\partial \psi}{\partial x}\frac{\partial w_i}{\partial x}\,\mathrm{d}x = \left[w_i\left(u^{n-2}\frac{\partial u}{\partial x} + u^n\right)\right]_{a(t)}^{b(t)} - \int_{a(t)}^{b(t)} \frac{\partial w_i}{\partial x}\left[u^{n-2}\frac{\partial u}{\partial x} + u^n\right]\,\mathrm{d}x.$$

The right-hand side simplifies, due to the zero boundary conditions, giving

$$\int_{a(t)}^{b(t)} u\frac{\partial \psi}{\partial x}\frac{\partial w_i}{\partial x}\,\mathrm{d}x = -\int_{a(t)}^{b(t)} \frac{\partial w_i}{\partial x}\left[u^{n-2}\frac{\partial u}{\partial x} + u^n\right]\,\mathrm{d}x, \tag{5.50}$$

which, for known u, can be used to obtain the velocity potential ψ.

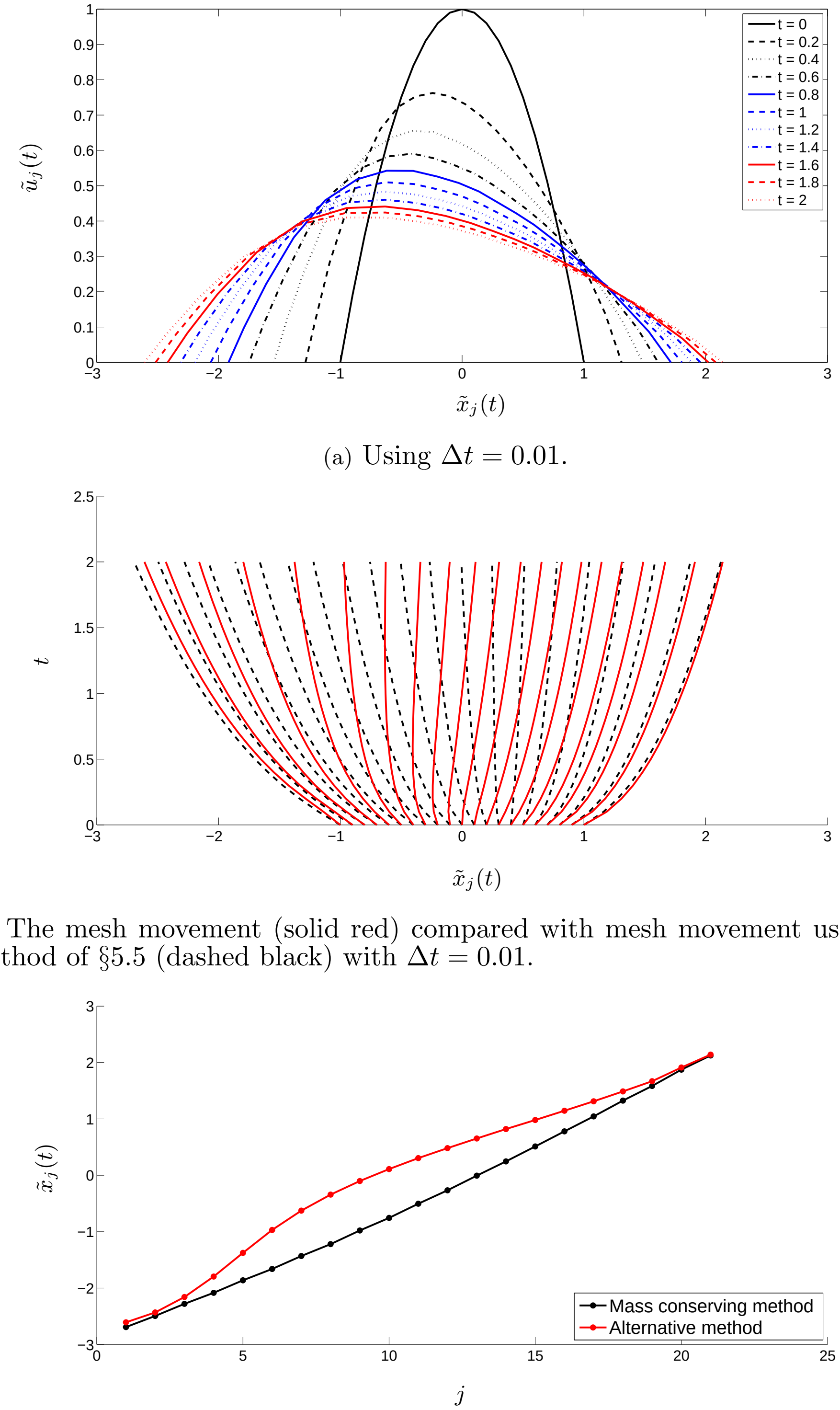
Next, the velocity $v(x,t)$ is calculated from a one-dimensional form of (2.20),

$$\int_{a(t)}^{b(t)} w_i v\,\mathrm{d}x = -\int_{a(t)}^{b(t)} w_i\frac{\partial \psi_i}{\partial x}\,\mathrm{d}x. \tag{5.51}$$

The velocity is used with a time-stepping scheme to update the mesh x.

To find the solution $u(x,t)$ we take the one-dimensional form of (2.21),

$$\int_{a(0)}^{b(0)} w_i(x,0)u(x,0)\,\mathrm{d}x = \int_{a(t)}^{b(t)} w_i(x,t)u(x,t)\,\mathrm{d}x. \tag{5.52}$$



(a) Using $\Delta t = 0.01$.

(b) The mesh movement (solid red) compared with mesh movement using the method of §5.5 (dashed black) with $\Delta t = 0.01$.

(c) The position of the nodes at $t = 2$.

Fig. 5.5: Richards' equation using the alternative moving mesh method given in §5.6, with $\Delta t = 0.02$, $n = 3$, $N = 20$.

5.8.1 Numerically solving Richards' equation using finite elements

We solve the equations for the finite element velocity potential (5.50), the finite element velocity (5.51), and the finite element solution (5.52) as systems of equations using piecewise linear expansions, $\Psi = \sum \Psi_j \phi_j$, $U = \sum U_j \phi_j$, $V = \sum V_j \phi_j$, which we substitute into each equation in turn, with $w_i = \phi_i$.

The equation for the finite element velocity potential (5.50) becomes

$$\int_{a(t)}^{b(t)} U \left[\sum_{j=0}^{N} \Psi_j \frac{\partial \phi_j}{\partial x} \right] \frac{\partial \phi_i}{\partial x} \, \mathrm{d}x = - \int_{a(t)}^{b(t)} \frac{\partial \phi_i}{\partial x} \left[U^{n-2} \sum_{j=0}^{N} U_j \frac{\partial \phi_j}{\partial x} + U^n \right] \mathrm{d}x, \tag{5.53}$$

where U, Ψ and ϕ are piecewise linear forms of u, ψ and w respectively. In (5.53) it is convenient to substitute the summation for U_x, but not U. Interchanging the summation and integral gives

$$\sum_{j=0}^{N} \Psi_j \left[\int_{a(t)}^{b(t)} \frac{\partial \phi_j}{\partial x} \frac{\partial \phi_i}{\partial x} U \, \mathrm{d}x \right] = - \sum_{j=0}^{N} U_j \left[\int_{a(t)}^{b(t)} \frac{\partial \phi_j}{\partial x} \frac{\partial \phi_i}{\partial x} U^{n-2} \, \mathrm{d}x \right] - \int_{a(t)}^{b(t)} \frac{\partial \phi_i}{\partial x} U^n \, \mathrm{d}x, \tag{5.54}$$

for $i = 0, 1, \ldots, N$. In [5]

$$K_{ij}(U) = \int_{a(t)}^{b(t)} U \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} \, \mathrm{d}x, \qquad S_i(U) = \int_{a(t)}^{b(t)} \frac{\partial \phi_i}{\partial x} U^n \, \mathrm{d}x,$$

are defined. Using these definitions (5.54) can be written

$$\sum_{j=0}^{N} K_{ij}(U) \Psi_j = - \sum_{j=0}^{N} K_{ij}(U^{n-2}) U_j - S_i(U). \tag{5.55}$$

Hence, to determine the mesh velocity potential at time $t = t^m$ we solve the matrix system

$$K(U)\mathbf{\Psi} = -K(U^{n-2})\mathbf{U} - \mathbf{S}(U), \tag{5.56}$$

where $K(U^n) = K(U^n)^m$ is the stiffness matrix defined by (5.55), $\mathbf{S}(U) = \mathbf{S}(U)^m$ is the vector of $S_i(U)$ values, $\mathbf{\Psi} = \mathbf{\Psi}^m$ is the vector $(\Psi_1^m, \ldots, \Psi_N^m))^T$, and $\mathbf{U} = \mathbf{U}^m$ is the vector $(U_1^m, \ldots, U_N^m))^T$. We solve (5.56) to find the velocity potential vector $\mathbf{\Psi}$.

By the same process, and from the equation for the finite element velocity (5.51), they determine that the mesh velocity at $t = t^m$ is given by

$$M\mathbf{V} = -H\mathbf{\Psi}, \tag{5.57}$$

where $M = M^m$ is the mass matrix defined by

$$M_{ij} = \int_{a(t)}^{b(t)} \phi_i \phi_j \,\mathrm{d}x, \tag{5.58}$$

$H = H^m$ is defined by

$$H_{ij} = \int_{a(t)}^{b(t)} \phi_i \frac{\partial \phi_j}{\partial x} \,\mathrm{d}x,$$

and $\mathbf{V} = \mathbf{V}^m$ is the vector $(V_1^m, \dots, V_N^m))^T$. We solve (5.57) to find the velocity vector $\mathbf{V}$.

The updated mesh x_j^{m+1} is determined by using V_j^m at $t = t^m$ with the explicit Euler time-stepping scheme

$$x_j^{m+1} = x_j^m + \Delta t V_j^m, \qquad j = 1, \dots, N-1. \tag{5.59}$$

Similarly, by substituting the expansions into the equation for the solution (5.52) we find the solution at $t = t^{m+1}$ by solving

$$M^{m+1}\mathbf{U}^{m+1} = \mathbf{g}, \tag{5.60}$$

where the vector

$$\mathbf{g} = M^0\mathbf{U}^0, \tag{5.61}$$

is determined from initial conditions, and the updated mass matrix M^{m+1} is calculated from the new mesh. Note that the distributed integrals $\int \phi_i U \,\mathrm{d}x$ are preserved in time.

In summary, the general algorithm for finite element moving mesh approach for Richards' equation is

Preliminary: Determine $\mathbf{g}$ from the initial conditions using (5.61).

Step 1: Calculate the velocity potential $\mathbf{\Psi}^m$ by solving (5.56);

Step 2: Calculate the mesh velocity $\mathbf{V}^m$ by solving (5.57);

Step 3: Use an explicit time-stepping scheme, such as (5.59), to update the mesh;

Step 4: Use the new mesh to calculate the new mass matrix M^{m+1} from (5.58);

Step 5: Find the updated solution by solving (5.60).

We now take a closer look at the implementation of this algorithm.

5.8.2 Numerical details

We provide specifics to the general algorithm given in the the last section to solve Richards' equation. We use nodes x_j^m, $j = 0, 1, \ldots, N$, and time-step Δt.

Preliminaries

To begin we determine the constant vector $\mathbf{g}$ from (5.60) using the initial mesh and initial conditions. Using an initial equispaced mesh we calculate the initial mass matrix M^0 which is a tridiagonal matrix with lower, main and upper diagonals, $\boldsymbol{Ml}^m$, $\boldsymbol{Md}^m$ and $\boldsymbol{Mu}^m$ given by

$$\begin{aligned}
\boldsymbol{Ml}^m &= \left[\cdots\cdots\left\{\frac{1}{6}\Delta x_{j_-}^m\right\}\cdots\cdots\left\{\frac{1}{6}\big(x_N^m - x_{N-1}^m\big)\right\}\right], && (5.62)\\
\boldsymbol{Md}^m &= \left[\left\{\frac{1}{3}(x_1^m - x_0^m)\right\}\cdots\left\{\frac{1}{3}(\Delta x_{j_-}^m + \Delta x_{j_+}^m)\right\}\cdots\left\{\frac{1}{3}(x_N^m - x_{N-1}^m)\right\}\right], && (5.63)\\
\boldsymbol{Mu}^m &= \left[\left\{\frac{1}{6}(x_1^m - x_0^m)\right\}\cdots\cdots\left\{\frac{1}{6}\Delta x_{j_+}^m\right\}\cdots\cdots\right], && (5.64)
\end{aligned}$$

for $m = 0$, where $j = 0, 1, \ldots, N$, $\Delta x_{j_-}^m = (x_j^m - x_{j-1}^m)$ and $\Delta x_{j_+}^m = (x_{j+1}^m - x_j^m)$. The initial $\mathbf{U}^0$ is given by sampling the initial solution at the nodes. Once $\mathbf{g}$ is determined we begin the time-stepping process.

Moving the mesh

To solve (5.56) for the velocity potential ψ at each time-level, we require the stiffness matrices $K(U)^m$ and $K(U^{n-2})^m$ which are tridiagonal matrices with lower, main and upper diagonals $\boldsymbol{Kl}^m$, $\boldsymbol{Kd}^m$ and $\boldsymbol{Ku}^m$ given by

$$\begin{aligned}
\boldsymbol{Kl}^m &= \left[\cdots\cdots\left\{-\frac{(u^n)_{j-\frac{1}{2}}^m}{\Delta x_{j_-}^m}\right\}\cdots\cdots\left\{-\frac{(u^n)_{N-1}^m}{x_N^m - x_{N-1}^m}\right\}\right],\\
\boldsymbol{Kd}^m &= \left[\left\{\frac{(u^n)_0^m}{x_1^m - x_0^m}\right\}\cdots\left\{\frac{(u^n)_{j-\frac{1}{2}}^m}{\Delta x_{j_-}^m} + \frac{(u^n)_{j+\frac{1}{2}}^m}{\Delta x_{j_+}^m}\right\}\cdots\left\{\frac{(u^n)_{N-1}^m}{x_N^m - x_{N-1}^m}\right\}\right],\\
\boldsymbol{Ku}^m &= \left[\left\{-\frac{(u^n)_0^m}{x_1^m - x_0^m}\right\}\cdots\cdots\left\{-\frac{(u^n)_{j+\frac{1}{2}}^m}{\Delta x_{j_+}^m}\right\}\cdots\cdots\right],
\end{aligned}$$

where $j = 0, 1, \ldots, N$, $\Delta x_{j_+}^m = (x_{j+1}^m - x_j^m)$ and $\Delta x_{j_-}^m = (x_j^m - x_{j-1}^m)$. The stiffness matrices $K(U)^m$ and $K(U^{n-2})^m$ are substituted into (5.56) to determine the velocity potential $\boldsymbol{\Psi}$.

The velocity potential $\mathbf{\Psi}^m$ is substituted into (5.57) to give the mesh velocity $\mathbf{V}^m$

$$\mathbf{V}^m = (M^m)^{-1} H^m \mathbf{\Psi}^m, \tag{5.65}$$

where the vector $H^m\mathbf{\Psi}^m$ is evaluated as

$$\mathbf{\Psi}^m = \left(\frac{1}{2}\left(\Psi_1^m - \Psi_0^m\right), \ldots, \frac{1}{2}\left(\Psi_{j+1}^m - \Psi_{j-1}^m\right), \ldots, \frac{1}{2}\left(\Psi_N^m - \Psi_{N-1}^m\right), \right)^T.$$

where $j = 0, 1, \ldots, N$. The mesh velocity $\mathbf{V}^m = (V_0^m, V_1^m, \ldots, V_N^m)^T$ is used in the explicit Euler time-stepping scheme to determine the new mesh points

$$x_j^{m+1} = x_j^m + \Delta t V_j^m.$$

Recovery of the solution

To recover the solution $\mathbf{U}^{m+1}$ we solve

$$M^{m+1}\mathbf{U}^{m+1} = \mathbf{g}\mathbf{\Psi}^m, \tag{5.66}$$

from (5.60), where the tridiagonal matrix M^{m+1} is defined by the vectors (5.62)–(5.64) using the new mesh values x_j^{m+1}.

A summary

A summary of the procedure to solve Richards' equation is:

Preliminary: Determine the initial mass matrix M^0 defined by the vectors (5.62)–(5.64). Using the initial M^0 and initial $\mathbf{U}^0$, calculate $\mathbf{g}$ from (5.61);

Step 1(a): Form the two $N \times N$ stiffness matrices, $K(U^n))$ and $K(U)$;

Step 1(b): Calculate the velocity potential ψ from (5.55);

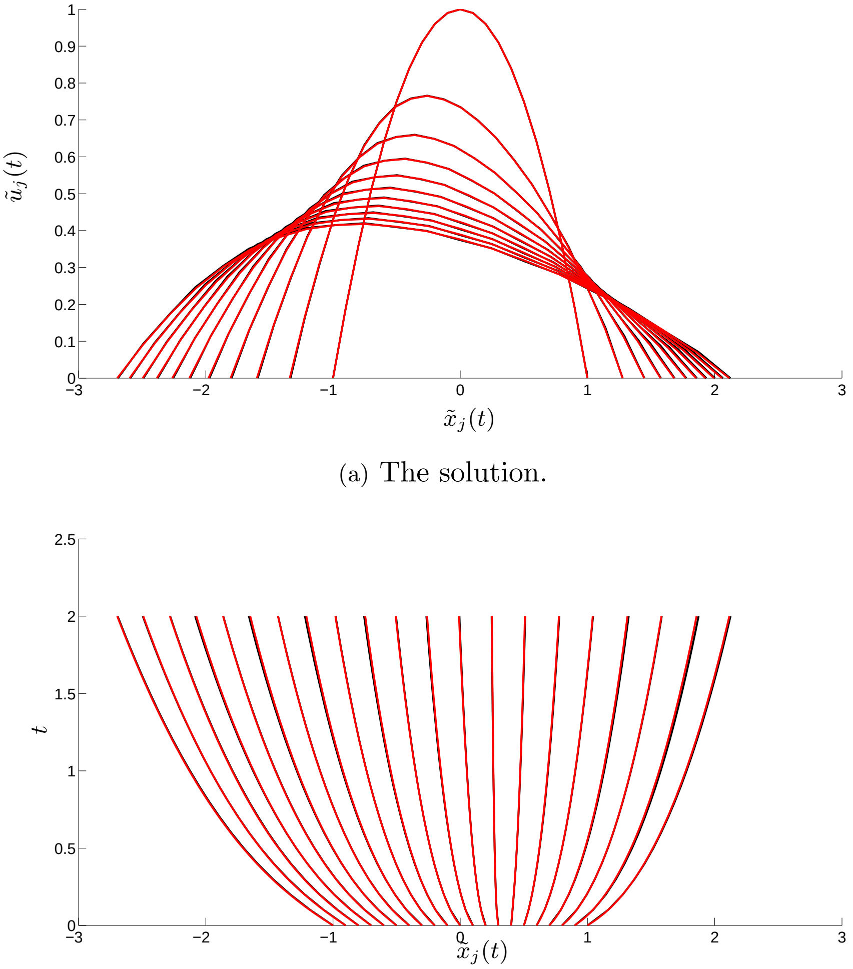
Step 2: Calculate the mesh velocity $V^m = (V_1^m, V_2^m, \ldots, V_N^m)^T$ from (5.65);

Step 3: Find the updated mesh from (5.59);

Step 4: Determine the new $N \times N$ mass matrix, M^{m+1} from (5.62)–(5.64);

Step 5: Find the updated solution from (5.66).

Figure 5.6(a) shows that the finite element method and finite difference method produce very similar results. Since the finite element approach and finite difference approach use



(a) The solution.

(b) The movement of the nodes in time.

Fig. 5.6: Richards' equation solved with finite elements (red) and finite differences (black), $N = 20$, $\Delta t = 0.01$.

the same motivation to define the mesh velocity, it is expected that the mesh nodes move similarly, see Figure 5.6(b).

In the next section we summarise our work on Richards' equation.

5.9 Summary for Richards' equation

In this chapter we solved Richards' equation numerically. Richards' equation is a fluid flow problem, so as with the PME, using a velocity-based moving mesh method is an elegant approach to model the solution since the nodes follow the flow itself (because the solution conserves mass). Although, this is not the case when the mesh is moved by balancing the partial masses with the flux term.

We showed that the mass of the solution remains constant, and that the centre of mass moves in one direction. Knowing that the centre of mass moves in one direction allows us to check that our numerical solution is exhibiting the expected behaviour.

We observed that deriving a self-similar solution for Richards' equation which satisfies the boundary conditions is not straightforward. This provides motivation for a numerical approach. We used conservation of partial masses with explicit and semi-implicit time-stepping. However, the semi-implicit scheme did not satisfy Theorem 3.4.1. As an alternative we used a method that balanced the partial masses with the flux term, leading to a semi-implicit time-stepping method such that Theorem 3.4.1 was satisfied. To investigate the accuracy we compared our numerical results from both approaches with results from a very fine, fixed mesh. We found that our conservation of mass approach was more accurate than the alternative approach, with a higher rate of convergence.

We also used the Conservation Method of Baines, Hubbard and Jimack [5] to solve Richards' equation. Since [5] does not include numerically solving Richards' equation, we have provided details about the numerical procedure and calculations. We found our finite difference approach where we conserve mass fractions, and the original finite element approach achieved very similar results.

Having considered two problems which conserve mass, we now consider a more general problem, the Crank-Gupta problem which does not conserve mass.

6
The Crank-Gupta Problem

6.1 Introduction

In [38] Crank and Gupta introduced the so-called oxygen-consumption problem for the evolution of oxygen concentration in a tissue, in which oxygen is absorbed at a prescribed constant rate. Oxygen is allowed to diffuse into a medium, and some of the oxygen is absorbed by the medium, thereby being removed from the diffusion process. The resulting diffusion-with-absorption process is represented by the non-dimensionalised PDE

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - 1, \tag{6.1}$$

where $u(x, t)$ denotes the concentration of the oxygen free to diffuse at a distance x from the outer surface of the medium at time t [38]. By considering the steady state Crank and Gupta [38] found that the outer boundary satisfies

$$u = \frac{\partial u}{\partial x} = 0 \quad \text{at } x = b(t). \tag{6.2}$$

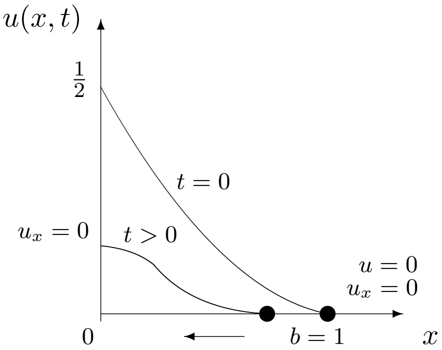


Fig. 6.1: Diagrammatic representation of the Crank-Gupta solution.

Oxygen in the region $0 \leq x \leq b(t)$ is consumed, causing the boundary $b(t)$ to recede toward $x = 0$. Crank and Gupta [38] showed that this leads to the additional boundary condition

$$\frac{\partial u}{\partial x} = 0 \quad \text{at } x = 0. \tag{6.3}$$

This model has a decreasing mass (due to the negative source term), as shown in Figure 6.1.

In this chapter we first seek a self-similar solution for the Crank-Gupta equation. However, we find that our series solution does not satisfy the outer boundary condition so we cannot compare our numerical solution to it: this nicely demonstrates the need for a numerical method. We go on to apply our moving mesh method so as to conserve *relative* mass fractions, as described in §3.2. We also apply the moving mesh method to a two-dimensional radially symmetric version of the Crank-Gupta problem, in §6.4. We compare our numerical results to results achieved using a Fourier Series approach [39] to give an indication of the accuracy of our moving mesh method when applied to problems that do not conserve mass. To allow a direct comparison to an exact solution we apply the moving mesh method to the Crank-Gupta PDE with a different inner boundary condition, provided by [5], which has an exact solution.

6.2 A self-similar solution

6.2.1 Scale Invariance

We consider the scaling transformation

$$t = \lambda \hat{t}, \qquad x = \lambda^{\beta} \hat{x}, \qquad u = \lambda^{\gamma} \hat{u}, \tag{6.4}$$

where λ is the scaling parameter and β and γ are constants.

Transforming the derivatives of (6.1) into the variables $\hat{t}$, $\hat{x}$ and $\hat{u}$ gives

$$\begin{aligned}\frac{\partial u}{\partial t} &= \lambda^{\gamma-1}\frac{\partial \hat{u}}{\partial \hat{t}},\\ \frac{\partial^2 u}{\partial x^2} &= \lambda^{\gamma-2\beta}\frac{\partial^2 \hat{u}}{\partial \hat{x}^2}.\end{aligned}$$

Hence, our transformed PDE (6.1) is

$$\lambda^{\gamma-1}\frac{\partial \hat{u}}{\partial \hat{t}} = \lambda^{\gamma-2\beta}\frac{\partial^2 \hat{u}}{\partial \hat{x}^2} - \lambda^0.$$

Therefore, for equation (6.1) to be invariant under the transformation (6.4) we require $\gamma - 1 = \gamma - 2\beta = 0$, hence

$$\beta = \frac{1}{2} \qquad \text{and} \qquad \gamma = 1, \tag{6.5}$$

so the scale invariant transformation (6.4) becomes

$$t = \lambda\hat{t}, \qquad x = \lambda^{\frac{1}{2}}\hat{x}, \qquad u = \lambda\hat{u}. \tag{6.6}$$

To summarise, the variables u, x and t can be rescaled as in (6.6) for any value of λ, whilst still satisfying the Crank-Gupta problem (6.1).

We now define scale-invariant similarity variables. From (6.6) we have

$$\lambda = \frac{t}{\hat{t}} = \frac{x^{\frac{1}{\beta}}}{\hat{x}^{\frac{1}{\beta}}} = \frac{u^{\frac{1}{\gamma}}}{\hat{u}^{\frac{1}{\gamma}}}.$$

Bearing this rescaling in mind, we introduce the two new variables

$$\zeta = \frac{u}{t^\gamma} = \frac{\hat{u}}{\hat{t}^\gamma}, \tag{6.7}$$

$$\xi = \frac{x}{t^\beta} = \frac{\hat{x}}{\hat{t}^\beta}, \tag{6.8}$$

which are independent of λ and are scale invariant under the transformation (6.4). We use (6.7) and (6.8) to seek a self-similar solution for the Crank-Gupta problem (6.1) in the next subsection.

6.2.2 Self-Similar Solutions

We derive a series self-similar solution for the Crank-Gupta PDE. We are not aware of any literature which already presents this. Suppose that ζ is a function of ξ and transform the

PDE (6.1) into the variables ζ and ξ to obtain an ODE. The left-hand side of (6.1) is the same as the PME and Richards' equation, thus the transformed left-hand side is (4.21) (given in §4.4),

$$\frac{\partial u}{\partial t} = -\beta t^{\gamma-1}\xi\frac{\mathrm{d}\zeta}{\mathrm{d}\xi} + \zeta(\xi)\gamma t^{\gamma-1}. \tag{6.9}$$

Transforming the right-hand side of (6.1) in a similar manner gives

$$\begin{aligned}\frac{\partial^2 u}{\partial x^2} - 1 &= \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right) - 1,\\ &= \frac{\partial \xi}{\partial x}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\frac{\partial \xi}{\partial x}\frac{\partial u}{\partial \zeta}\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}\right) - 1.\end{aligned}$$

Substituting $\frac{\partial \xi}{\partial x} = t^{-\beta}$ and $\frac{\partial u}{\partial \zeta} = t^{\gamma}$ from (6.7)–(6.8), in the first term on the right-hand side,

$$\begin{aligned}\frac{\partial^2 u}{\partial x^2} - 1 &= t^{-\beta}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(t^{\gamma-\beta}\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}\right) - 1,\\ &= t^{\gamma-2\beta}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}\right) - 1. \end{aligned} \tag{6.10}$$

Putting together (6.9) and (6.10) gives the Crank-Gupta PDE (6.1) in terms of ζ and ξ for general β and γ,

$$-\beta t^{\gamma-1}\xi\frac{\mathrm{d}\zeta}{\mathrm{d}\xi} + \zeta(\xi)\gamma t^{\gamma-1} = t^{\gamma-2\beta}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\frac{\mathrm{d}\zeta}{\mathrm{d}\xi}\right) - 1.$$

Note that t disappears from the equation since $\gamma - 1 = \gamma - 2\beta = 0$ (from (6.5)). Substituting β and γ from (6.5), gives the ODE

$$\frac{\mathrm{d}^2\zeta}{\mathrm{d}\xi^2} + \frac{\xi}{2}\frac{\mathrm{d}\zeta}{\mathrm{d}\xi} - \zeta(\xi) = 1. \tag{6.11}$$

The solution of this ODE, along with the previous definitions $u = \zeta t^{\gamma}$ and $x = \xi t^{\beta}$ provides the self-similar solution.

To solve (6.11) we make the substitution

$$\zeta = s - 1, \tag{6.12}$$

resulting in the homogeneous ODE

$$\frac{\mathrm{d}^2 s}{\mathrm{d}\xi^2} + \frac{\xi}{2}\frac{\mathrm{d}s}{\mathrm{d}\xi} - s(\xi) = 0, \tag{6.13}$$

where $\xi \in [0,1]$.

To find a solution to (6.13) we consider a series solution of the form

$$s = \sum_{k=0}^{\infty} c_k \xi^k, \tag{6.14}$$

where c_k are constants to be determined. By substituting (6.14), and its derivatives, into (6.13) we achieve the equation

$$2c_2 - c_0 + \sum_{k=1}^{\infty} \left[(k+2)(k+1)c_{k+2} + \frac{k}{2}c_k - c_k \right] \xi^k = 0. \tag{6.15}$$

For (6.15) to hold, each coefficient of ξ^k, $k = 0, 1, \ldots$ must be zero, hence

$$\begin{array}{rclc} \text{for } k & = & 0, & 2c_2 - c_0 = 0, \\ \text{for } k & = & 1,2,3\ldots, & (k+2)(k+1)c_{k+2} + c_k\left(\dfrac{k}{2} - 1\right) = 0. \end{array}$$

Thus,

$$\begin{aligned} c_2 &= \frac{c_0}{2}, &(6.16)\\ c_{k+2} &= \frac{c_k\left(1 - \frac{k}{2}\right)}{(k+2)(k+1)}. &(6.17) \end{aligned}$$

Substituting $k = 2$ into (6.17) gives

$$c_4 = \frac{c_2\left(1 - \frac{2}{2}\right)}{3 \cdot 4} = 0.$$

Thus, by induction, $c_k = 0$ for even $k \geq 4$. Substituting for odd values of k gives

$$\begin{array}{rcll} k & = & 1: & c_3 = \dfrac{c_1}{2 \cdot 2 \cdot 3} = \dfrac{c_1}{2 \cdot 3!}, \\ k & = & 3: & c_5 = \dfrac{c_3\left(1 - \frac{3}{2}\right)}{4 \cdot 5} = -\dfrac{c_3}{2 \cdot 4 \cdot 5} = -\dfrac{c_1}{2^2 \cdot 5!}, \\ k & = & 5: & c_7 = \dfrac{c_5\left(1 - \frac{5}{2}\right)}{6 \cdot 7} = -\dfrac{3 \cdot c_5}{2 \cdot 6 \cdot 7} = \dfrac{3 \cdot c_1}{2^3 \cdot 7!}, \\ k & = & 7: & c_9 = \dfrac{c_7\left(1 - \frac{7}{2}\right)}{8 \cdot 9} = -\dfrac{5 \cdot c_7}{2 \cdot 8 \cdot 9} = -\dfrac{3 \cdot 5 \cdot c_1}{2^4 \cdot 9!}, \\ k & = & 9: & c_{11} = \dfrac{c_9\left(1 - \frac{9}{2}\right)}{10 \cdot 11} = -\dfrac{7 \cdot c_9}{2 \cdot 10 \cdot 11} = \dfrac{3 \cdot 5 \cdot 7 \cdot c_1}{2^5 \cdot 11!}, \\ k & = & 11: & c_{13} = \dfrac{c_{11}\left(1 - \frac{11}{2}\right)}{12 \cdot 13} = -\dfrac{9 \cdot c_{11}}{2 \cdot 10 \cdot 11} = -\dfrac{3 \cdot 5 \cdot 7 \cdot 9 \cdot c_1}{2^6 \cdot 13!}. \end{array}$$

Let $k = 2m + 1$, $m = 1, 2, 3, \ldots$, such that for $m = 1, 3, 4, 5, 6$ ($k = 3, 5, 7, 9, 11$) we observe that

$$c_{2m+1} = \frac{(-1)^{m+1}}{2^{2m-1}} \cdot \frac{(2m-2)!}{(2m+1)!(m-1)!} \cdot c_1, \tag{6.18}$$

and we write (6.17) as

$$c_{2m+3} \;=\; \frac{(1-2m)}{2(2m+3)(2m+2)} c_{2m+1}. \tag{6.19}$$

To prove that (6.18) holds for general m we substitute it into (6.19),

$$\begin{aligned} c_{2m+3} &= \frac{(-1)^{m+1}}{2^{2m-1}} \cdot \frac{(1-2m)}{2(2m+3)(2m+2)} \cdot \frac{(2m-2)!}{(2m+1)!(m-1)!} \cdot c_1, \\ &= \frac{(-1)^{m+2}(2m-1)}{2^{2m}(2m+3)(2m+2)} \cdot \frac{1}{(2m+1)(2m)(2m-1)(m-1)!} \cdot c_1, \\ &= \frac{(-1)^{m+2}}{2^{2m+1}(2m+3)(2m+2)(2m+1)m!} \cdot c_1, \\ &= \frac{(-1)^{m+2}(2m)!}{2^{2m+1}(2m+3)!m!} \cdot c_1, \end{aligned}$$

which corresponds to (6.18) for index $(2m + 3)$, hence (6.18) holds for all $m = 1, 2, 3, \ldots$, by induction. Therefore, a solution to (6.13) is

$$s = c_0 \;+\; c_1\xi + \frac{c_0}{2}\xi^2 + c_1 \sum_{m=1}^{\infty} \frac{(-1)^{m+1}}{2^{2m-1}} \cdot \frac{(2m-2)!}{(2m+1)!(m-1)!}\xi^{2m+1}. \tag{6.20}$$

To ensure that (6.20) is a valid series solution to (6.13), we investigate whether the series converges by using the alternating series test to determine whether

$$a_m = (-1)^{m+1} \cdot \frac{(2m-2)!}{2^{2m-1}(2m+1)!(m-1)!} =: (-1)^{m+1} b_m, \qquad m = 2, 3, 4 \ldots,$$

converges. Ergo, if both

(1) $\lim_{m=\infty} b_m = 0$;

(2) b_m is a decreasing sequence;

then a_m converges.

To prove (1) we take b_m,

$$\begin{aligned} b_m &= \frac{(2m-2)!}{2^{2m-1}(2m+1)!(m-1)!}, \\ &= \frac{1}{2^{2m-1}(2m+1)(2m)(2m-1)(m-1)!}. \end{aligned}$$

We immediately observe that as $m \to \infty$, $b_m \to 0$.

To prove (2) we note that

$$\frac{b_{m+1}}{b_m} = \frac{2^{2m-1}(2m+1)!(2m)!(m-1)!}{2^{2m+1}(2m+3)!(2m-2)!m!} = \frac{2m(2m-1)}{4m(2m+3)(2m+2)}.$$

Since $4m(2m+3)(2m+2) > 2m(2m-1)$,

$$\frac{b_{m+1}}{b_m} < 1,$$

hence b_m is a decreasing sequence.

We have confirmed that (6.20) is a converging series solution to (6.13), so we use the definition (6.12) to give a series solution of the ODE (6.11)

$$\zeta(\xi) = -1 + c_0 + c_1\xi + \frac{c_0}{2}\xi^2 + c_1 \sum_{m=1}^{\infty} \frac{(-1)^{m+1}}{2^{2m-1}} \cdot \frac{(2m-2)!}{(2m+1)!(m-1)!}\xi^{2m+1}. \tag{6.21}$$

From (6.3), $\zeta_\xi = 0$ at $\xi = 0$ giving $c_1 = 0$,. Hence

$$\zeta(\xi) = -1 + c_0 + \frac{c_0}{2}\xi^2.$$

Finally, to achieve the self-similar solution in terms of u and x we substitute for ζ and ξ using the definitions (6.7)–(6.8) for β and γ defined by (6.5),

$$u(x,t) = -t + c_0 t + \frac{c_0}{2}x^2, \tag{6.22}$$

which is easily verified to satisfy (6.1). At $\xi = 1$, the outer boundary, $\zeta = \zeta_\xi = 0$ from (6.2), so we cannot satisfy the other two boundary conditions using (6.21). However, if we shift x to $(1-x)$ the solution (6.22) becomes

$$u(x,t) = -t + c_0 t + \frac{c_0}{2}(1-x)^2,$$

which is also a solution of (6.1) that satifies (6.2), if $c_0 = 1$. This is plotted in Figure 6.2 but (6.3) cannot be satisfied, and regardless, this solution does not depend on t.

Since we were unable to satisfy all the boundary conditions with a realistic series solution we compared our numerical results with those from a similar problem that has a different boundary condition which does have an analytical solution. Further details are given in §6.5.

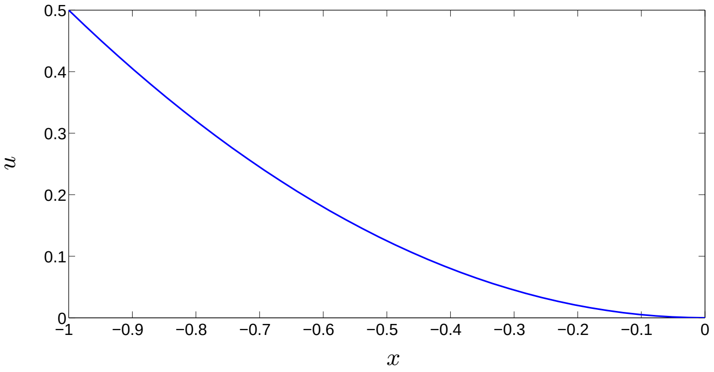


Fig. 6.2: A self-similar solution for the Crank-Gupta problem, $u = \frac{1}{2}(1-x)^2$.

We have seen that deriving an analytical solution for the Crank-Gupta problem is not straightforward, which provides motivation to seek a numerical solution. In the next section we apply our moving mesh method to the Crank-Gupta problem.

6.3 Moving mesh method

Since the mass in the Crank-Gupta problem is not conserved in time we use the moving mesh method described in §3.2, with the same notation, i.e. $\tilde{x}_j(t^m) \approx x_j^m$ denotes the jth node of the mesh with $N+1$ nodes, at time $m\Delta t$, $m = 0, 1 \ldots$, and $u_j^m \approx \tilde{u}_j(t^m)$ and $v_j^m \approx \tilde{v}_j(t^m)$ denote the solution and mesh velocity at these nodes. The total mass of the solution at $m\Delta t$ is $\theta^m \approx \theta(t^m)$.

We model the Crank-Gupta problem by (3.17) with

$$\mathcal{G}u \equiv \frac{\partial^2 u}{\partial x^2} - 1, \tag{6.23}$$

where u satisfies the boundary conditions (6.2)–(6.3), which we repeat here,

$$\begin{aligned} \frac{\partial u}{\partial x} &= 0 \qquad \text{at} \qquad x = a(t) = 0, && (6.24) \\ u = \frac{\partial u}{\partial x} &= 0 \qquad \text{at} \qquad x = b(t), && (6.25) \end{aligned}$$

where $b(t)$ is a moving boundary.

We show that given a mesh $\tilde{x}_j(t^m)$, with corresponding solution $\tilde{u}_j(t^m)$ and total mass $\theta(t^m)$, we can calculate the updated total mass $\theta(t^{m+1})$, mesh $\tilde{x}_j(t^{m+1})$ and solution $\tilde{u}_j(t^{m+1})$ by computing the rate of change of total mass $\dot{\theta}(t^m)$ and mesh velocity $\tilde{v}_j(t^m)$.

We use the initial condition

$$u(x,0) = \frac{1}{2}(1-x)^2, \tag{6.26}$$

for $x \in [0,1]$ as in the original paper [38]. Substituting the initial conditions (6.26) into equation (3.19), the initial total mass $\theta(0)$ is

$$\theta(0) = \frac{1}{2}\int_0^1 (1-x)^2 \,\mathrm{d}x = \frac{1}{6}. \tag{6.27}$$

To determine the normalised partial integrals c_j we substitute (6.26) and (6.27) into (3.20), giving

$$c_j \quad = \quad 6\int_0^{\tilde{x}_j(0)} \frac{1}{2}(1-x)^2 \,\mathrm{d}x = \tilde{x}_j(0)^3 - 3\tilde{x}_j(0)^2 + 3\tilde{x}_j(0),$$

which remain constant in time.

6.3.1 Determining the rate of change of total mass

The total mass $\theta(t)$ is updated using an explicit time-stepping scheme, where $\dot{\theta}$ is given by substituting the PDE (6.23) and the boundary conditions (6.24)–(6.25) into (3.21), giving

$$\dot{\theta}(t^m) = \int_0^{b(t)} \left\{ \frac{\partial^2 u}{\partial x^2} - 1 \right\} \mathrm{d}x = \left[\frac{\partial u}{\partial x} - x \right]_0^{b(t)} = -b(t). \tag{6.28}$$

The discrete form is simply

$$\dot{\theta}^m = -x_N^m, \tag{6.29}$$

where $\dot{\theta}^m \approx \dot{\theta}(t^m)$.

The new total mass θ^{m+1} is obtained from $\dot{\theta}^m$ by a time-stepping scheme.

6.3.2 Determining the mesh velocity

The total mass θ is updated with the mesh, where the mesh velocity is given by substituting (6.23)–(6.25) into (3.23), and evaluating the integral, so that

$$\tilde{v}_j(t) \quad = \quad \frac{1}{\tilde{u}_j(t)}\left(\dot{\theta}(t)c_j - \int_0^{\tilde{x}_j(t)}\left\{\frac{\partial^2 u}{\partial x^2} - 1\right\}\,\mathrm{d}x\right),$$

for interior points $j = 1, \ldots, N-1$. Substituting for $\dot{\theta}(t)$ from (6.28) gives

$$\tilde{v}_j(t) \quad = \quad -\frac{1}{\tilde{u}_j(t)}\left(c_j b(t) + \left.\frac{\partial u}{\partial x}\right|_{\tilde{x}_j(t)} - \tilde{x}_j(t)\right). \tag{6.30}$$

We use a discretised form of (6.30),

$$v_j^m = -\frac{1}{u_j^m}\left[c_j x_N^m + \left(\frac{u_{j+1}^m - u_{j-1}^m}{x_{j+1}^m - x_{j-1}^m}\right) - x_j^m\right], \qquad j = 1, 2, ..., N-1. \tag{6.31}$$

We note that the term in the curved brackets can be replaced by (4.37) (where $n = 1$), to give a discretisation of (6.30) which is more accurate on a non-uniform mesh, namely

$$v_j^m = -\frac{1}{u_j^m}\left[c_j x_N^m + \left(\frac{\frac{1}{\Delta x_{j+}^m}\left(\frac{\Delta u_{j+}^m}{\Delta x_{j+}^m}\right) + \frac{1}{\Delta x_{j-}^m}\left(\frac{\Delta u_{j-}^m}{\Delta x_{j-}^m}\right)}{\frac{1}{\Delta x_{j+}^m} + \frac{1}{\Delta x_{j-}^m}}\right) - x_j^m\right], \tag{6.32}$$

for interior nodes $j = 1, 2, \ldots, N-1$, where $\Delta(\cdot)_{j-}^m = (\cdot)_j^m - (\cdot)_{j-1}^m$ and $\Delta(\cdot)_{j+}^m = (\cdot)_{j+1}^m - (\cdot)_j^m$. The new mesh x_j^{m+1}, $j = 1, \ldots, N-1$, is obtained from v_j^m by a time-stepping scheme.

At the outer boundary $u(b,t) = 0$ from (6.25), so we seek an alternative method to determine x_N^m. One approach is to extrapolate the boundary velocity v_N^m from the internal velocities, and then update the position of the outer node along with the internal nodes. However, extrapolation sometimes produces a numerical solution with a boundary that moves out (the boundary should move in [38]). As an alternative, we consider the asymptotic behaviour of the solution near the outer boundary. At the outer boundary $u = 0$ so $u_t = 0$, reducing (6.1) to

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=b} \quad = \quad 1.$$

The Taylor expansion for $u(x)$ about $x = b$ is

$$u(x) = u(b) + (b-x)\left.\frac{\partial u}{\partial x}\right|_{x=b} + \frac{(b-x)^2}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=b} + \ldots,$$

hence,

$$u(x,t) \approx \frac{1}{2}(\tilde{x}_j - b(t))^2, \tag{6.33}$$

close to $b(t)$. Therefore, for the discrete case where $j = N-1$ and $t = t^{m+1}$, we make the approximation

$$u_{N-1}^{m+1} \approx \frac{1}{2}(x_{N-1}^{m+1} - x_N^{m+1})^2,$$

which gives the following formula for the outer node,

$$x_N^{m+1} = x_{N-1}^{m+1} + \sqrt{2u_{N-1}^{m+1}}, \tag{6.34}$$

taking the positive square root.

6.3.3 Recovering the solution

Once the updated mesh x_j^{m+1} has been determined, the updated solution u_j^{m+1}, $j = 1,\ldots,N-1$, is given by either (3.25) for a uniform mesh, or (3.26) for a non-uniform mesh. The solution at the inner boundary u_0^{m+1} is calculated using

$$u_0^{m+1} = \frac{\theta^{m+1}}{\theta^0}\frac{x_1^0 u_0^0}{x_1^{m+1}},$$

from (3.25) and the boundary condition (6.24). At the outer boundary, $u_{N+1}^{m+1} = 0$ from (6.25).

6.3.4 The full algorithm

Given a total mass θ^m, mesh x_j^m, solution u_j^m, $j = 0,\ldots,N$, at t^m, $m \geq 0$:

- Compute the rate of change of mass $\dot{\theta}^m$ from (6.29);
- Compute the mesh velocity v_j^m from (6.31) or (6.32);
- Compute the updated mesh x_j^{m+1} by a time-stepping scheme;
- Compute the updated solution u_j^{m+1} from (3.25) or (3.26).

6.3.5 Time-stepping schemes

Explicit schemes

The simplest method to time-step the mesh is the first order explicit Euler time-stepping scheme,

$$\frac{x_j^{m+1} - x_j^m}{\Delta t} = v_j^m,$$

for $j = 1, \ldots, N-1$. We substitute for v_j^m from (6.31) or (6.32). At the inner boundary $v_0^m = 0$, and at the outer boundary we use (6.34). The explicit Euler time-stepping scheme requires small Δt so that the x_j^m remain stable, and to avoid mesh tangling. We also implemented the adaptive predictor-correcter Runge-Kutta methods in Matlab. We used the solver, ODE15s, which is designed to solve a stiff system. Implementing this solver produced results similar to results from ODE23 and ODE45 (which are not designed for stiff systems), inferring that the method does not lead to a stiff system.

A semi-implicit scheme

The semi-implicit scheme described in §3.4.2 updates the mesh x_j^m only. Thus, for a method that requires the total mass θ to be updated as well, we cannot use it in the same way, since θ would have to be updated separately (for example, using the explicit Euler time-stepping scheme). By not updating the mesh and mass simultaneously, the maximum principle for monotonicity is lost. Consequently, when implementing our moving mesh method we use only explicit time-stepping schemes.

We have given details of applying the moving mesh method to the one-dimensional Crank-Gupta problem. We now demonstrate that the same method can be applied to the two-dimensional radially symmetric Crank-Gupta problem.

6.4 The two-dimensional radially symmetric case

Our moving mesh method described in §3.2 is easily generalised to numerically solve the two-dimensional radially symmetric Crank-Gupta problem, which is determined by converting the general form of (6.1)

$$\frac{\partial u}{\partial t} = \nabla^2 u - 1,$$

to polar coordinates using $r^2 = x^2 + y^2$ giving

$$\frac{\partial u}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) - 1, \tag{6.35}$$

with boundary conditions

$$\frac{\partial u}{\partial r} = 0 \qquad \text{at} \qquad r = 0,\, t > 0, \tag{6.36}$$

$$u = 0,\ \frac{\partial u}{\partial r} = 0 \quad \text{at} \quad R(t),\, t > 0, \tag{6.37}$$

and initial conditions

$$u = \frac{1}{2}(1-r)^2, \qquad r \in [0,1],\, t = 0. \tag{6.38}$$

As with the one-dimensional case, we introduce the dependent variable $\tilde{r}_j(t)$, $j = 0, ..., N$, to represent the $N+1$ nodes on the radius of the mesh, which are dependent on t. The mesh is initially equally-spaced. We define the velocity of the j-th node $s(\tilde{r}_j, t)$ on the radius to be

$$s(\tilde{r}_j, t) = \tilde{s}_j(t) = \frac{\mathrm{d}\tilde{r}}{\mathrm{d}t}.$$

We assume conservation of relative mass (as with the one-dimensional case) such that

$$d_j = \frac{1}{\psi(t)}\int_0^{\tilde{r}_j(t)} u(r,t)r\ \mathrm{d}r, \tag{6.39}$$

where d_j is a constant in time, $\psi(t)$ is the total mass

$$\psi(t) = \int_0^{R(t)} u(r,t)r\ \mathrm{d}r, \tag{6.40}$$

and $R(t)$ is the outer boundary. Specifically, for initial conditions (6.38),

$$\psi(0) = \int_0^1 (1-r)^2 r\ \mathrm{d}r = \frac{7}{12},$$

and

$$d_j = \frac{12}{7}\int_0^{\tilde{r}_j(0)} (1-r)^2 r\ \mathrm{d}r = \frac{12}{7}r_j(0)^2 - \frac{8}{7}r_j(0)^3 + \frac{3}{7}r_j(0)^4.$$

Given a mesh $\tilde{x}_j(t)$ and solution $\tilde{u}_j(t) = u(\tilde{x}_j(t), t)$, we can evaluate the total mass $\psi(t)$ directly from (6.40). To evaluate an updated value of the total mass (which is required for determining the updated solution) we compute $\dot{\psi}(t)$, and then approximate the total mass using a time-stepping scheme. Simultaneously, the mesh velocity $\tilde{s}_j(t)$ is computed, and the mesh and total mass are updated together. This ultimately enables us to recover the updated solution on the new mesh. Details are given in the following subsections.

6.4.1 Determining the rate of change of total mass

To determine an expression for the total mass $\psi(t)$, we first calculate the rate of change of the total mass $\dot{\psi}(t)$ by applying the Leibnitz integral rule to the right-hand side of (6.40),

$$\dot{\psi}(t) = \int_0^{R(t)} \frac{\partial u}{\partial t} r \, \mathrm{d}r + u(R,t)R(t)\frac{\partial R}{\partial t}.$$

Substituting in $\frac{\partial u}{\partial t}$ from (6.35), and eliminating the last term due to the boundary condition (6.37),

$$\dot{\psi}(t) = \int_0^{R(t)} \left\{ \frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) - r \right\} \mathrm{d}r.$$

Hence, using the boundary conditions (6.36)–(6.37),

$$\dot{\psi}(t) = -\int_0^{R(t)} r \, \mathrm{d}r = -\frac{R(t)^2}{2}.$$

We use the discrete form,

$$\dot{\psi}^m = -\frac{(r^2)_N^m}{2}, \tag{6.41}$$

where $\dot{\psi}^m \approx \dot{\psi}(t^m)$.

The total mass is determined at each time-level, together with the updated mesh, using a time-stepping scheme.

6.4.2 Determining the mesh velocity

To find an expression for the mesh velocity, we differentiate (6.39) with respect to time using the Leibnitz integral rule, giving

$$\dot{\psi}(t)d_j = \frac{\mathrm{d}}{\mathrm{d}t}\int_0^{\tilde{r}_j(t)} u(r,t)r \, \mathrm{d}r = \int_0^{\tilde{r}_j(t)} \frac{\partial u}{\partial t} r \, \mathrm{d}r + \tilde{u}_j(t)\tilde{r}_j(t)\tilde{s}_j(t) - \tilde{u}_j(0)\tilde{r}_j(0)\tilde{s}_j(0).$$

Substituting for $\frac{\partial u}{\partial t}$ from (6.35), and cancelling the last term due to the boundary condition (6.36),

$$\dot{\psi}(t)d_j \quad = \quad \int_0^{\tilde{r}_j(t)} \left\{ \frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial t}\right) - r \right\} \, \mathrm{d}r + \tilde{u}_j(t)\tilde{r}_j(t)\tilde{s}_j(t).$$

Evaluating the integral and using the boundary condition (6.36),

$$\dot{\psi}(t)d_j \quad = \quad \tilde{r}_j(t)\left.\frac{\partial u}{\partial r}\right|_{\tilde{r}_j(t)} - \frac{\tilde{r}_j(t)^2}{2} + \tilde{u}_j(t)\tilde{r}_j(t)\tilde{s}_j(t).$$

Hence, the radially symmetric Crank-Gupta mesh velocity is given by

$$s(\tilde{r}_j, t) = \frac{1}{\tilde{u}_j(t)\tilde{r}_j(t)}\left(\dot{\psi}(t)d_j - \tilde{r}_j(t)\left.\frac{\partial u}{\partial r}\right|_{\tilde{r}_j(t)} + \frac{\tilde{r}_j(t)^2}{2}\right), \tag{6.42}$$

for $\tilde{u}_j(t) \neq 0$ and $\tilde{r}_j(t) \neq 0$. In practice we have used a one-sided average for $\tilde{u}_j(t)\tilde{r}_j(t)$, and a one-sided approximation to discretise $\frac{\partial u}{\partial r}\big|_{\tilde{r}_j(t)}$, so (6.42) becomes

$$s_j^m \quad = \quad \frac{4}{(u_j^m + u_{j-1}^m)(r_j^m + r_{j-1}^m)}\left\{ \dot{\psi}^m d_j - r_j^m \left(\frac{u_j^m - u_{j-1}^m}{r_j^m - r_{j-1}^m}\right) + \frac{(r^2)_j^m}{2} \right\}, \tag{6.43}$$

which holds for $j = N$ and hence for $j = 1, \ldots, N$. By definition, $s_0^m = 0$. The new mesh r_j^{m+1} is obtained from s_j^m by a time-stepping scheme.

Unfortunately, the averaged approximation for $\tilde{u}_j(t)\tilde{r}_j(t)$ results in the behaviour of the solution at the outer boundary being poorly represented in the numerical solution. Hence we seek an alternative method to determine r_N^m. A polynomial extrapolation for s_N^m from the internal velocities sometimes produces a numerical solution with a boundary that moves out, as with the one-dimensional case. In §6.3.2 we derived the approximation (6.34) by considering the asymptotic behaviour of u near the boundary (where u_t is small). A similar approach in the radial case gives

$$1 = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) = \frac{\partial^2 u}{\partial r^2}$$

since $\frac{\partial u}{\partial t} = \frac{\partial u}{\partial r} = 0$ at $r = R$. At $x = b$ this leads to

$$u(r, t) \approx \frac{1}{2}(\tilde{r}_j - R(t))^2,$$

via the Taylor series. This is the same as the Cartesian case (6.33), therefore we can define the outer node by

$$r_N^{m+1} = r_{N-1}^{m+1} + \sqrt{2u_{N-1}^{m+1}}, \tag{6.44}$$

taking the positive square root.

6.4.3 Recovering the solution

To approximate the updated solution u_j^{m+1}, we equate (6.39) at times $t = t^{m+1}$ and $t = 0$ between the points $\tilde{r}_{j+1}$ and $\tilde{r}_{j-1}$,

$$\frac{1}{\psi(t^{m+1})} \int_{r_{j-1}^{m+1}}^{r_{j+1}^{m+1}} u(r, t^{m+1}) r \, \mathrm{d}r = \frac{1}{\psi(0)} \int_{r_{j-1}^{0}}^{r_{j+1}^{0}} u(r, 0) r \, \mathrm{d}r.$$

Approximating the integrals by the mid-point rule,

$$u_j^{m+1} = \frac{\psi^{m+1}}{\psi^0} \frac{\left(r_{j+1}^0 - r_{j-1}^0\right) r_j^0}{\left(r_{j+1}^{m+1} - r_{j-1}^{m+1}\right) r_j^{m+1}} u_j^0, \tag{6.45}$$

for $j = 1, \ldots, N-1$, where $u_N^{m+1} + 0$ from (6.37). For $j = 0$

$$u_0^{m+1} = \frac{\psi^{m+1}}{\psi^0} \frac{r_1^0}{r_1^m} u_j^0, \tag{6.46}$$

where we have applied the boundary condition (6.36).

6.4.4 The full algorithm

Given a total mass θ^m, mesh r_j^m, and solution u_j^m, $j = 0, \ldots, N$, at t^m, $m \geq 0$:

- Compute the rate of change of mass $\dot{\theta}^m$ from (6.41);
- Compute the mesh velocity s_j^m from (6.43);
- Compute the updated mesh x_j^{m+1} by a time-stepping scheme;
- Compute the updated solution u_j^{m+1} from (6.45).

We have given details of applying our moving mesh method to the original Crank-Gupta problem for the one-dimensional case and the two-dimensional, radially symmetric case. There is no known analytical solution for the Crank-Gupta problem. In order to compare our results to an exact solution we now consider the Crank-Gupta PDE with a different boundary condition such that there is a known exact solution which we can use for comparison.

6.5 Alternative boundary conditions

There is no analytic solution to the Crank-Gupta problem. However, the one-dimensional Cartesian Crank-Gupta problem *without* the inner boundary conditions (6.3) imposed has an exact solution

$$u = \begin{cases} e^{x+t-1} - x - t & x \leq 1-t, \\ 0 & x > 1-t, \end{cases} \tag{6.47}$$

as given in [5], corresponding to a new inner boundary condition,

$$\frac{\partial u}{\partial x} = e^{t-1} - 1 \qquad \text{at} \quad x = 0, \tag{6.48}$$

replacing (6.3). The initial conditions are given by the exact solution (6.47) at $t = 0$,

$$u = e^{x-1} - x \qquad \text{at} \quad t = 0, \tag{6.49}$$

for $x \in [0, 1]$. By applying our moving mesh method to this modified Crank-Gupta model, we can investigate the accuracy of the method for this problem.

Substituting the initial conditions (6.49) into equation (3.19) gives the initial total mass $\theta(0)$,

$$\theta(0) = \int_0^1 e^{x-1} - x \, \mathrm{d}x = \frac{1}{2} - e^{-1}. \tag{6.50}$$

To determine the normalised partial integrals c_j we substitute (6.49) and (6.50) into (3.20),

$$\begin{aligned} c_j &= \frac{1}{\frac{1}{2} - e^{-1}} \int_0^{\tilde{x}_j(0)} \left(e^{x-1} - x \right) \mathrm{d}x, \\ &= \frac{e^{-1}(e^{\tilde{x}_j(0)} - 1) - \frac{\tilde{x}_j^2(0)}{2}}{\frac{1}{2} - e^{-1}}, \\ &= \frac{2e^{-1}(e^{\tilde{x}_j(0)} - 1) - \tilde{x}_j^2(0)}{1 - 2e^{-1}}. \end{aligned}$$

We use the same notation given in §6.3 and follow the same process to find an updated solution on the new mesh. We use the explicit Euler time-stepping scheme for this problem.

6.5.1 Determining the rate of change of total mass

The total mass θ is updated using an explicit time-stepping scheme, where $\dot{\theta}$ is given by (6.1), (6.3) and (6.48) substituted into (3.21),

$$\dot{\theta}(t) = \int_0^{b(t)} \left\{ \frac{\partial^2 u}{\partial x^2} - 1 \right\} \mathrm{d}x = 1 - e^{t-1} - b(t).$$

Comparing this to (6.28) from the original problem, we observe that $1 - e^{t-1}$ is the result of implementing the alternative boundary condition. The discrete form of this is simply

$$\dot{\theta}^m = 1 - e^{t^m - 1} - x_N^m, \tag{6.51}$$

where $\dot{\theta}^m \approx \dot{\theta}(t^m)$.

The total mass is determined at each time-level, together with the updated mesh, using a time-stepping scheme.

6.5.2 Determining the mesh velocity

The total mass θ is updated with the mesh, where the mesh velocity is given by (6.1), (6.3) and (6.48) substituted into (3.23), such that

$$\begin{aligned} \tilde{v}_j(t) &= \frac{1}{\tilde{u}_j(t)} \left(\dot{\theta}(t) c_j - \int_0^{\tilde{x}_j(t)} \left\{ \frac{\partial^2 u}{\partial x^2} - 1 \right\} \mathrm{d}x \right), \\ &= \frac{1}{u(x_j, t)} \left(c_j \dot{\theta}(t) - \left. \frac{\partial u}{\partial x} \right|_{\tilde{x}_j(t)} + \tilde{x}_j(t) + e^{t-1} - 1 \right), \end{aligned} \tag{6.52}$$

for the interior points $j = 0, 1, \ldots, N-1$. Comparing to the mesh velocity from the original problem (6.30), we observe that the additional $(e^{t-1} - 1)$ terms are the result of implementing the alternative boundary condition. When discretising u_x in (6.52) we can use a mid-point approximation, which is first order accurate on a non-uniform mesh,

$$v_j^m = \frac{1}{u_j^m} \left(c_j \dot{\theta}^m - \left(\frac{(u^n)_{j+1}^m - (u^n)_{j-1}^m}{(x_{j+1}^m - x_{j-1}^m)} \right) + x_j^m + e^{t^m - 1} - 1 \right). \tag{6.53}$$

Alternatively, for a second order approximation on a non-uniform mesh we refer to the approximation given by equation (4.37), with $n = 1$, such that we use the discrete form of (6.52)

$$v_j^m = \frac{1}{u_j^m}\left(c_j\dot{\theta}^m - \left(\frac{\frac{1}{\Delta x_{j+}^m}\left(\frac{\Delta u_{j+}^m}{\Delta x_{j+}^m}\right) + \frac{1}{\Delta x_{j-}^m}\left(\frac{\Delta u_{j-}^m}{\Delta x_{j-}^m}\right)}{\frac{1}{\Delta x_{j+}^m} + \frac{1}{\Delta x_{j-}^m}} \right) + x_j^m + e^{t^m - 1} - 1 \right), \qquad (6.54)$$

where $\Delta(\cdot)_{j-}^m = (\cdot)_j^m - (\cdot)_{j-1}^m$ and $\Delta(\cdot)_{j+}^m = (\cdot)_{j+1}^m - (\cdot)_j^m$, and $v_0^m = 0$ at the inner boundary. At the right boundary, $u(b,t) = 0$, as for the original Crank-Gupta problem, so we seek an alternative method to determine x_N^m. Again, either v_N^m is extrapolated from the internal velocities using a polynomial, or equation (6.34) is employed since the outer boundary is the same as with the original problem.

The new mesh x_j^{m+1} is obtained from v_j^m by a time-stepping scheme.

6.5.3 Recovering the solution

The solution is recovered in the same manner as with the original Crank-Gupta problem in Chapter 6.3, i.e. by either (3.25), for a uniform mesh, or (3.26), for a non-uniform mesh. The solution at the inner boundary u_0^{m+1} is calculated using a one-sided approximation of (3.25)

$$u_0^{m+1} = \frac{\theta^{m+1}}{\theta^0}\frac{(x_1^0 - x_0^0)u_0^0}{x_1^{m+1} - x_0^{m+1}}.$$

At the outer boundary, $u_{N+1}^{m+1} = 0$ from the zero boundary condition (6.25).

6.5.4 The full algorithm

Given a total mass θ^m, mesh r_j^m, and solution u_j^m, $j = 0, \dots, N$, at t^m, $m \geq 0$:

- Compute the rate of change of mass $\dot{\theta}^m$ from (6.51);
- Compute the mesh velocity v_j^m from (6.53) or(6.54);
- Compute the updated mesh x_j^{m+1} by a time-stepping scheme;
- Compute the updated solution u_j^{m+1} from (3.25) or (3.26).

The last case we consider is the original Crank-Gupta problem where we move the nodes to preserve partial mass balances, as in §3.3.

6.6 A partial mass balance moving mesh method

The Crank-Gupta PDE has a source term so we can use the moving mesh method described in §3.3, with the same notation, i.e. $\tilde{x}_j(t^m) \approx x_j^m$ denotes the jth node of the mesh with $N+1$ nodes, at time $m\Delta t$, $m = 0, 1\ldots$, and $u_j^m \approx \tilde{u}_j(t^m)$ and $v_j^m \approx \tilde{v}_j(t^m)$ denote the solution and mesh velocity at these nodes. The partial masses of the solution at $m\Delta t$ are $\Theta_j^m \approx \Theta_j(t^m)$.

We model the Crank-Gupta problem by the PDE (3.27) with

$$\mathcal{H}u \equiv \frac{\partial^2 u}{\partial x^2} \quad \text{and} \quad S(x,t) \equiv -1, \quad a(t) = 0 \leq x \leq b(t), \tag{6.55}$$

from (6.1). Hence, the mass balance relation (3.29) is

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_0^{\tilde{x}_j(t)} u(x,t)\, \mathrm{d}x = -\int_0^{\tilde{x}_j(t)} 1\, \mathrm{d}x.$$

This relation implies that given a mesh $\tilde{x}_j(t^m)$, with corresponding solution $\tilde{u}_j(t^m)$ and partial masses $\Theta_j(t^m)$, we can calculate the updated partial masses $\Theta_j(t^{m+1})$, mesh $\tilde{x}_j(t^{m+1})$ and solution $\tilde{u}_j(t^{m+1})$ by computing the rate of change of partial masses $\dot{\Theta}_j(t^m)$ and mesh velocity $\tilde{v}_j(t^m)$.

We use the same initial solution from [38], namely

$$u(x,0) = \frac{1}{2}(1-x)^2, \tag{6.56}$$

for $x \in [0,1]$. Substituting the initial conditions (6.56) into equation (3.30) gives the initial partial masses $\Theta_j(0)$,

$$\Theta_j(0) \quad = \quad \frac{1}{2}\int_0^{\tilde{x}_j(0)} (1-x)^2\, \mathrm{d}x = \frac{\tilde{x}_j(0)^3}{6} - \frac{\tilde{x}_j(0)^2}{2} + \frac{\tilde{x}_j(0)}{2}.$$

.

6.6.1 Determining the rate of change of partial masses

The partial masses $\Theta_j(t)$ are updated using an explicit time-stepping scheme, where $\dot{\Theta}_j(t)$ is given by $S(x,t)$ from (6.55) substituted into (3.31),

$$\dot{\Theta}_j(t) = -\int_0^{\tilde{x}_j(t)} \mathrm{d}x = -\tilde{x}_j(t). \tag{6.57}$$

The discrete approximation to (6.57) is simply

$$\dot{\Theta}_j^m = -x_j^m, \tag{6.58}$$

where $\dot{\Theta}_j^m \approx \dot{\Theta}_j(t^m)$.

The partial masses are determined at each time-level, together with the updated mesh, using a time-stepping scheme.

6.6.2 Determining the mesh velocity

The partial masses $\Theta_j(t)$ are updated with the mesh, where the mesh velocity is given by (6.55) and (6.3) substituted into (3.33), so that the mesh velocity is

$$\tilde{v}_j(t) \;=\; -\frac{1}{\tilde{u}_j(t)}\int_0^{\tilde{x}_j(t)} \frac{\partial^2 u}{\partial x^2}\,\mathrm{d}x = -\frac{1}{\tilde{u}_j(t)}\frac{\partial u}{\partial x}\bigg|_{\tilde{x}_j(t)}, \tag{6.59}$$

for the internal nodes, $j = 1, \ldots, N-1$. Comparing to (6.30), from the method that conserves relative partial masses in §6.3, we note that the mesh velocity does not explicitly depend upon the boundary position $b(t)$ nor initial data (in the form of the partial masses c_j).

We use a discretised form of (6.59), which is first order accurate on a non-uniform mesh,

$$v_j^m = -\frac{1}{u_j^m}\left(\frac{u_{j+1}^m - u_{j-1}^m}{x_{j+1}^m - x_{j-1}^m}\right), \qquad j = 1, 2, ..., N-1. \tag{6.60}$$

We note that the term in the curved brackets can be replaced by (4.37) (where $n = 1$), to give a discretisation of (6.59) which is more accurate on a non-uniform mesh, namely

$$v_j^m = -\frac{1}{u_j^m}\left(\frac{\frac{1}{\Delta x_{j+}^m}\left(\frac{\Delta u_{j+}^m}{\Delta x_{j+}^m}\right) + \frac{1}{\Delta x_{j-}^m}\left(\frac{\Delta u_{j-}^m}{\Delta x_{j-}^m}\right)}{\frac{1}{\Delta x_{j+}^m} + \frac{1}{\Delta x_{j-}^m}}\right), \tag{6.61}$$

for $j = 1, 2, ..., N-1$, where $\Delta(\cdot)_{j-}^m = (\cdot)_j^m - (\cdot)_{j-1}^m$ and $\Delta(\cdot)_{j+}^m = (\cdot)_{j+1}^m - (\cdot)_j^m$. At the inner boundary $v_0^m = 0$. At the right boundary $u(b,t) = 0$, as with the original Crank-Gupta problem, so we seek an alternative method to determine x_N^m. Again, either v_N^m is extrapolated from a polynomial using the internal velocities or equation (6.34) is employed.

The new mesh x_j^{m+1} is obtained from v_j^m by a time-stepping scheme.

6.6.3 Recovering the solution

The solution is updated using the approach given in §3.3, namely, by either (3.34) or (3.35), the latter being more accurate for a non-uniform mesh. At the outer boundary, $u_{N+1}^{m+1} = 0$ from (6.2). At the inner boundary we use the trapezoidal approximation

$$\Theta_0^{m+1} = \frac{1}{2}(u_1^{m+1} + u_0^{m+1})x_1^{m+1},$$

thus

$$u_0^{m+1} = \frac{2\Theta_0^{m+1}}{x_1^{m+1}} - u_1^{m+1}.$$

6.6.4 The full algorithm

Given the partial masses Θ^m, mesh x_j^m, and solution u_j^m, $j = 0, \dots, N$, at t^m, $m \geq 0$:

- Compute the rate of change of partial masses $\dot{\Theta}_j^m$ from (6.58);
- Compute the mesh velocity v_j^m from (6.60) or (6.61);
- Compute the updated mesh x_j^{m+1} by a time-stepping scheme;
- Compute the updated solution u_j^{m+1} from (3.34) or (3.35).

In the last four sections we have given the details for applying the moving mesh method to the Crank-Gupta problem. In the next section we present the numerical results.

6.7 Numerical results

We have looked at solving the Crank-Gupta problem

- in the one-dimensional case, using the moving mesh method which preserves partial mass fractions (see §6.3);
- in the radially symmetric case, using the moving mesh method which preserves partial mass fractions (see §6.4);
- in the one-dimensional case, using alternative boundary conditions and the moving mesh method which preserves partial mass fractions (see §6.5);
- in the one-dimensional case, using the moving mesh method which preserves partial mass balance (see §6.6).

We now present results for each of these numerical simulations described earlier in this chapter, where we start with a equispaced mesh and use the explicit Euler time-stepping scheme in all cases.

6.7.1 Preserving partial mass fractions

We used the second order approximations (6.32) to compute the mesh velocity, and (3.26) to compute the updated solution. The boundary position was calculated using (6.34) (from considering the asymptotic behaviour of the solution near the outer boundary).

Figure 6.3(a) shows the numerical solution at various times for $t \in [0, 0.19]$. We note that the solution is behaving as expected, the outer boundary is moving in, whilst the inner boundary is levelling out to satisfy the boundary condition.

There is no known analytical solution to the Crank-Gupta problem. As a comparison we use the work of Dahmardah and Mayers [39] who derive a Fourier Series solution. By comparing their results to earlier work [51], they conclude that their method is very accurate. To check whether the method converges as N increases and Δt decreases, we compare $\tilde{u}_0(0.1)$ and $\tilde{x}_N(0.1)$ to the results from [39] at $t = 0.1$ which are

$$\begin{aligned} \bar{u} &= u(0, 0.1) = 0.143177, && (6.62)\\ \bar{x} &= b(0.1) = 0.935018. && (6.63) \end{aligned}$$

We solve for $t \in [0, 0.1]$ and compute results for $N = 20 \times 2^{\hat{N}-1}$, $\hat{N} = 0, \ldots, 5$. We denote the solution $\tilde{u}_0(0.1)$ and boundary position $\tilde{x}_N(0.1)$ for a particular value of N by u_N and x_N respectively. To balance the spatial and temporal errors, and recalling that we have used explicit Euler time-stepping, we use $\Delta t = \mathcal{O}\left(\frac{1}{N^2}\right)$, precisely $\Delta t = \frac{1}{1600(4^{\hat{N}})}$. We anticipate that the pointwise errors $|\bar{u} - u_N|$ and $|\bar{x} - x_N|$ will decrease as $\hat{N}$ increases, for each $i = 0, \ldots, 10$.

As a measure of the errors, we calculate

$$E_N(u) = \sqrt{\frac{(\bar{u} - u_N)^2}{(\bar{u})^2}}, \qquad E_N(x_N) = \frac{\bar{x}_N - x_N}{\bar{x}_N},$$

for $\hat{N} = 0, \ldots, 5$ (i.e. $N = 20, 40, 80, 160, 320, 640$). We investigate the hypothesis that

$$E_N(u) \sim \frac{1}{N^p} \qquad \text{and} \qquad E_N(x_N) \sim \frac{1}{N^q}, \tag{6.64}$$

for large N, where p and q are the estimated orders of convergence. If (6.64) holds then we would expect that p_{2N} and q_{2N} defined by

$$p_{2N} = -\log_2\left(\frac{E_{2N}(u)}{E_N(u)}\right), \quad q_{2N} = -\log_2\left(\frac{E_{2N}(x_N)}{E_N(x)}\right).$$

would approach the constant values p and q as $N \to \infty$. Since each step of our scheme is second order in space and first order in time, and recalling that $\Delta t = \mathcal{O}\left(\frac{1}{N^2}\right)$, we might expect to see $p, q \approx 2$.

N	u_N	$E_N(u)$	p_N	x_N	$E_N(x_N)$	q_N
20	0.142721	3.185×10^{-3}	-	0.935385	3.925×10^{-4}	-
40	0.143040	9.569×10^{-4}	1.7	0.935120	1.091×10^{-4}	1.8
80	0.143141	2.514×10^{-4}	1.9	0.935043	2.674×10^{-5}	2.0
160	0.143168	6.286×10^{-5}	2.0	0.935024	6.417×10^{-6}	2.0
320	0.143175	1.397×10^{-5}	2.2	0.935019	1.069×10^{-6}	2.6
640	0.143176	6.984×10^{-6}	1.0	0.935018	0	-

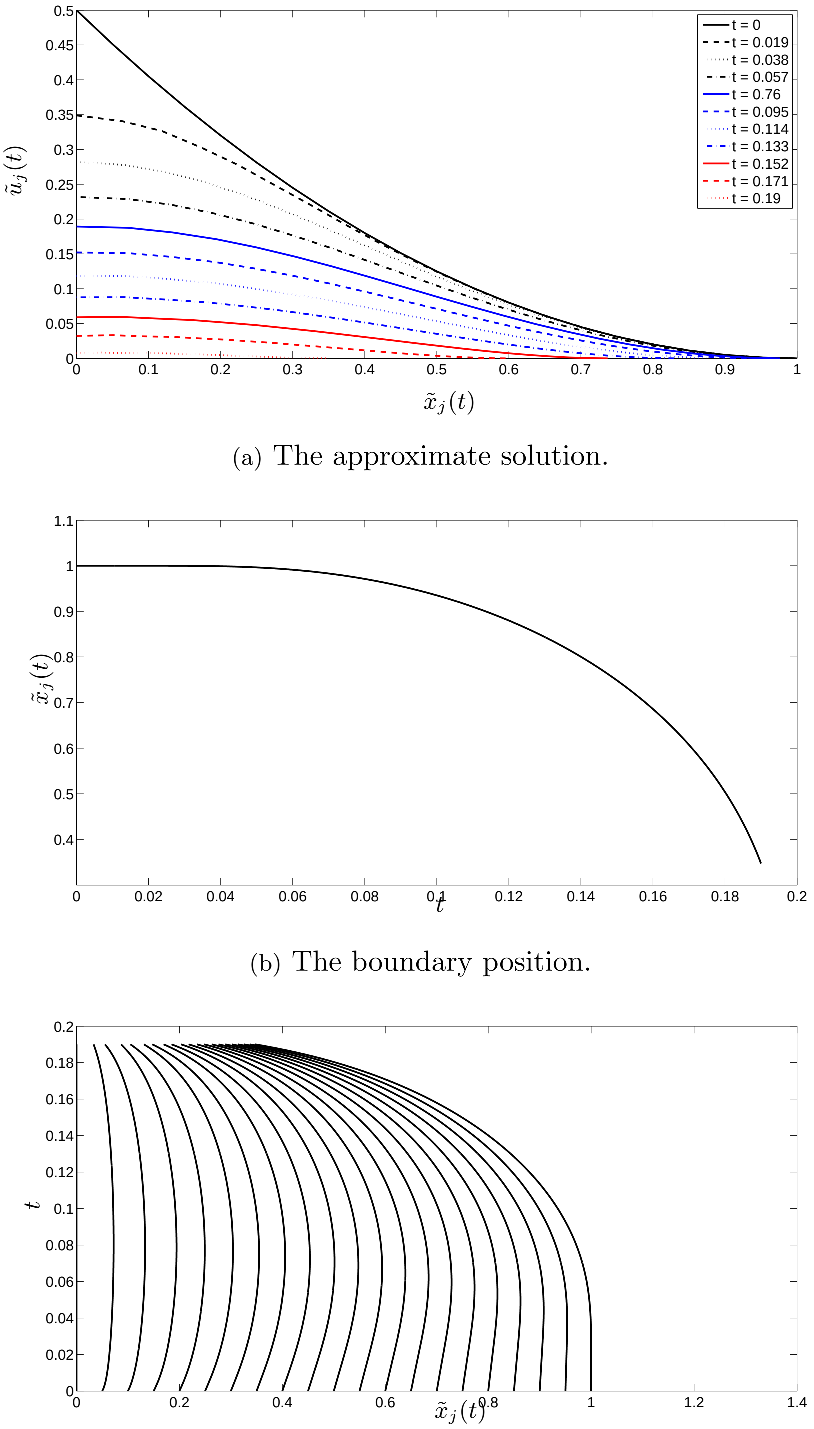
Table 6.1: Relative errors for u with rates of convergence using the explicit Euler time-stepping scheme.

There are some irregularities in Table 6.1, but it would be reasonable to suggest that the relative mass conserving moving mesh method, with explicit Euler time-stepping has second-order convergence. The irregular entries for the p values for $N = 640$ are most likely because we are comparing our results with a single value to the order of 10^{-6}. To this level of accuracy, our numerical results for $N = 640$ are very nearly identical to the Fourier Series results of [39], given by (6.62)–(6.63).

Further comparisons with the Fourier Series approach are given in Table 6.2, where we note that the results are very similar. The table also highlights that the moving mesh method gives a numerical boundary that moves out very slightly, before exhibiting the correct behaviour of moving in. The boundary behaviour movement for $t \in [0, 0.19]$ is shown in Figure 6.3(b).

Figure 6.3(c) shows the exact movement of 20 nodes over time. We observe that despite the inner boundary moving in, the nodes still cluster toward the outer boundary, where higher resolution allows greater accuracy to track the boundary movement.

In §6.3 we mentioned that updating the mesh velocities semi-implicitly, whilst updating the total mass θ explicitly, can lead to inaccuracies. This is shown in Figure 6.4 where the Crank-Gupta problem was solved with a semi-implicit scheme that satisfied Theorem 3.4.1. We note that the inner boundary condition $u_x = 0$ is not satisfied.



(a) The approximate solution.

(b) The boundary position.

(c) The mesh trajectory.

Fig. 6.3: The Crank-Gupta problem solved using relative partial mass conservation, $N = 20$, $\Delta t = 2 \times 10^{-4}$.

	Solution $u(0,t)$		Boundary $b(t)$	
t	Fourier Series	Moving mesh	Fourier Series	Moving mesh
0.01	0.387162	0.3943210	1.000000	1.0000011
0.02	0.340423	0.3446501	0.999999	0.9999885
0.03	0.304559	0.3074492	0.999911	0.9998163
0.04	0.274324	0.2763743	0.999180	0.9989005
0.05	0.247687	0.2491283	0.996793	0.9963278
0.10	0.143177	0.1428852	0.935018	0.9350185
0.12	0.109129	0.1084201	0.879171	0.8798560
0.14	0.077850	0.0768102	0.798944	0.8004555
0.16	0.048823	0.0475175	0.683449	0.6857932
0.18	0.021781	0.0202182	0.501329	0.5038886
0.19	0.009021	0.0071793	0.346000	0.3471498

Table 6.2: Comparing the Fourier Series approach with moving mesh results, $N = 20$, $\Delta t = 2 \times 10^{-4}$.

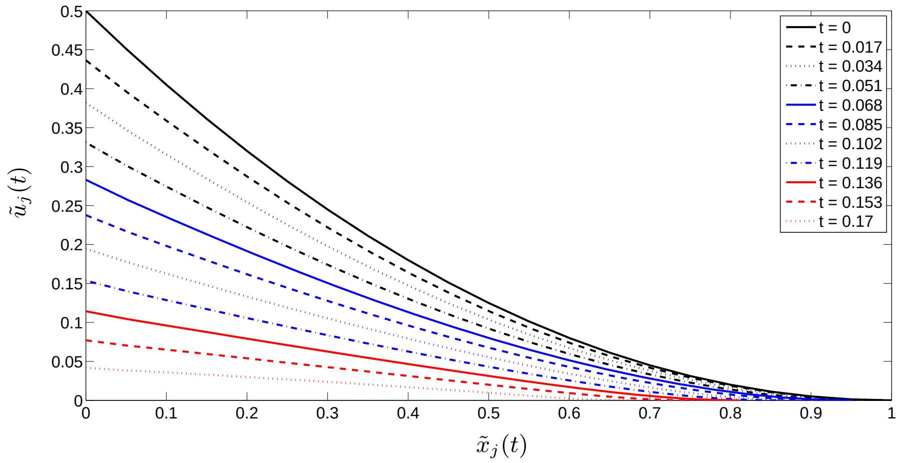


Fig. 6.4: The Crank-Gupta problem solved using relative partial mass conservation, with a semi-implicit time-stepping scheme, $N = 20$, $\Delta t = 4.25 \times 10^{-4}$.

6.7.2 The radially symmetric case

The radially symmetric case was solved using the moving method method which preserves partial mass fractions. We used (6.43) to compute the mesh velocity, and (6.45) and (6.46) to compute the updated solution. Despite using Laplacian smoothing on u_j we find that the solution is unsmooth at the inner boundary, for example see Figure 6.5(b).

Recall in §6.4.2 that we use an averaged approximation for $\tilde{u}_j(t)\tilde{r}_j(t)$ in (6.43) since (6.42) breaks down for $\tilde{u}_N(t) = 0$. This approach results in poor accuracy at the outer boundary, see Figures 6.5(b)–6.5(a). Furthermore, the boundary moves out considerably, not in, as required. The inaccuracy of boundary position is particularly noticeable in Figure 6.5(c) where it would be reasonable to suggest that the final node $\tilde{x}_{20}(t)$ should be mimicking the behaviour of $\tilde{x}_{19}(t)$.

We considered the asymptotic behaviour of u near the boundary (where u_t is small). Although this approach proved to be successful in the one-dimensional case, we observe from Figure 6.6 that approximating $\tilde{x}_N(t)$ using (6.44) gives poor boundary accuracy. We observe from Figure 6.6(c) that using (6.44) causes $\tilde{x}_{19}(t)$ to follow the inaccurate behaviour of $\tilde{x}_{20}(t)$, in addition $\tilde{x}_{20}(t)$ moves out more compared to Figure 6.5(c).

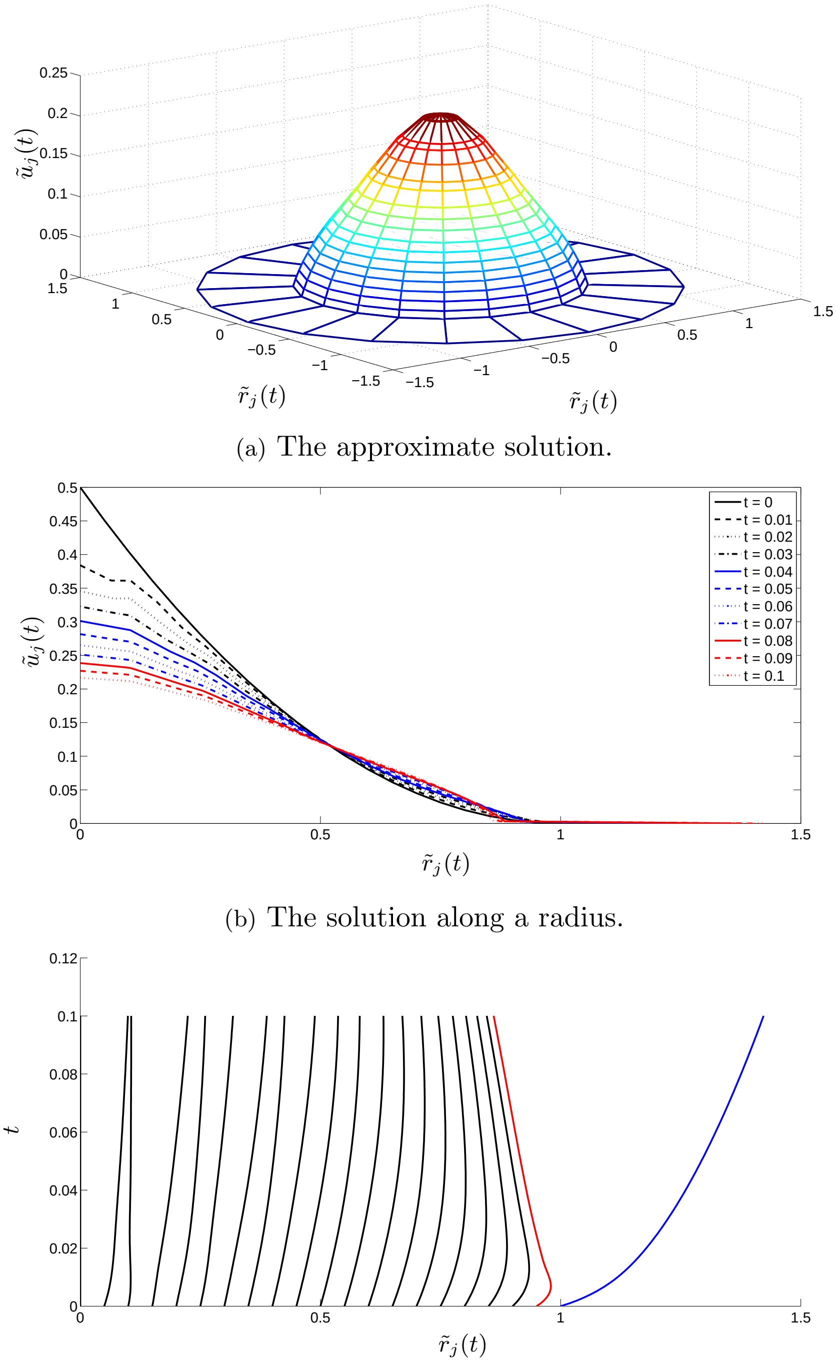
An alternative approach was to extrapolate the boundary velocity $\tilde{s}_N(t)$ from the nearby boundary velocities. Comparing this approach to the one-sided approximation approach we see that extrapolation gives a smoother solution at the boundary, see Figure 6.7. Figure 6.7(c) shows that the mesh nodes remain fairly equally spaced across most of the region, and the outer boundary moves out slightly less than Figure 6.5(c). However, the nodes at the inner boundary are closer than with Figures 6.5(c)–6.6(c), although the mesh remains monotonic.

We have incurred difficulties in achieving an accurate numerical solution at the boundaries. At the inner boundary, the nodes appears to gather slightly. As described, several different approaches have been used to improve on our numerical solution at the outer boundary, but unfortunately, we have not resolved this issue.

6.7.3 Alternative boundary conditions

The Crank-Gupta problem with alternative boundary conditions was solved using the moving method method which preserves partial mass fractions. We used the second order approximations (6.54) to compute the mesh velocity, and (3.26) to compute the updated solution.

As mentioned before, we were unable to compare the original Crank-Gupta problem to an analytical solution. However, by imposing an alternative boundary condition (6.48) we can examine convergence as N increases and Δt decreases over the whole region. We

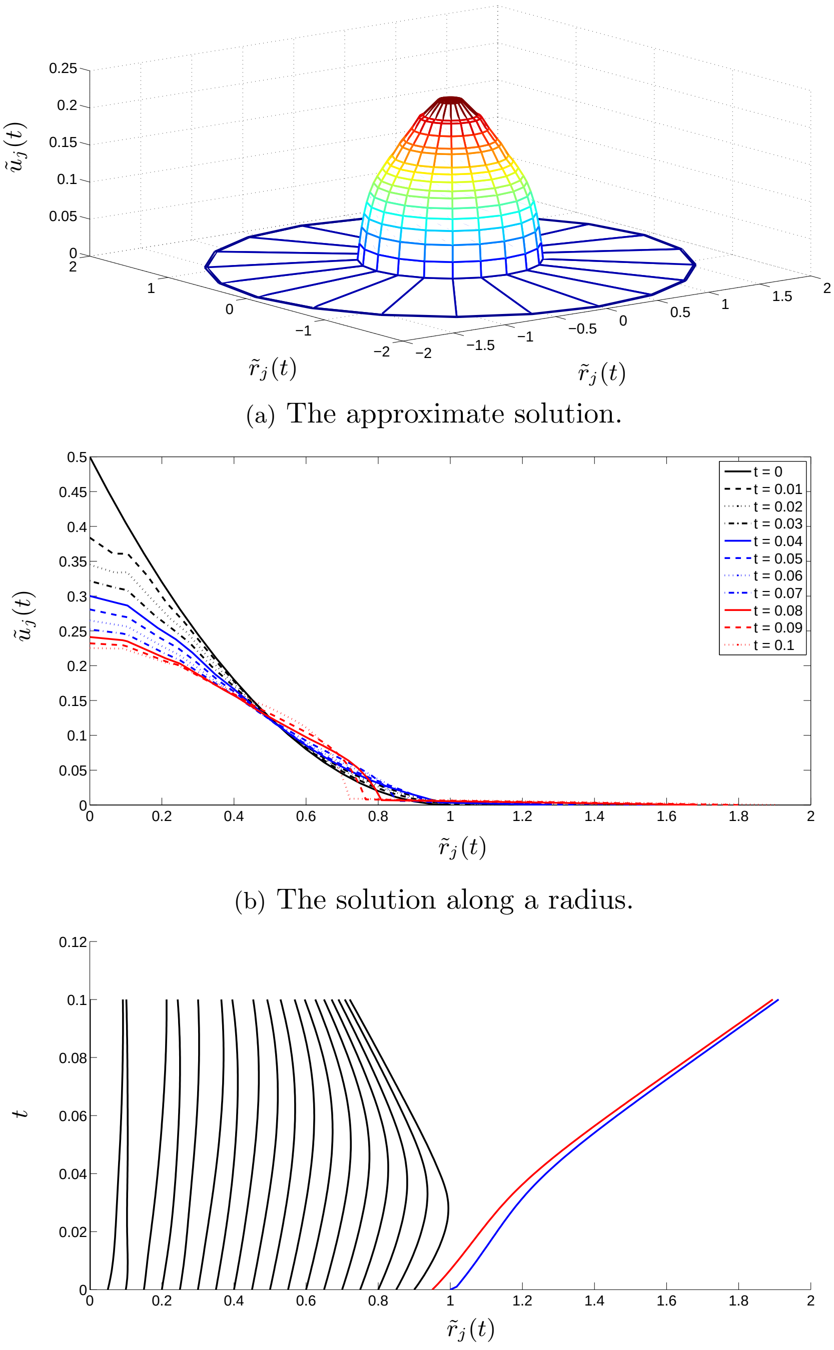


(a) The approximate solution.

(b) The solution along a radius.

(c) The mesh trajectory where $\tilde{r}_{19}(t)$ is red, and $\tilde{r}_{20}(t)$ is blue.

Fig. 6.5: The radial Crank-Gupta problem using equation (6.43) to approximate $\tilde{s}_N(t)$, $N = 20$, $\Delta t = 1 \times 10^{-4}$.

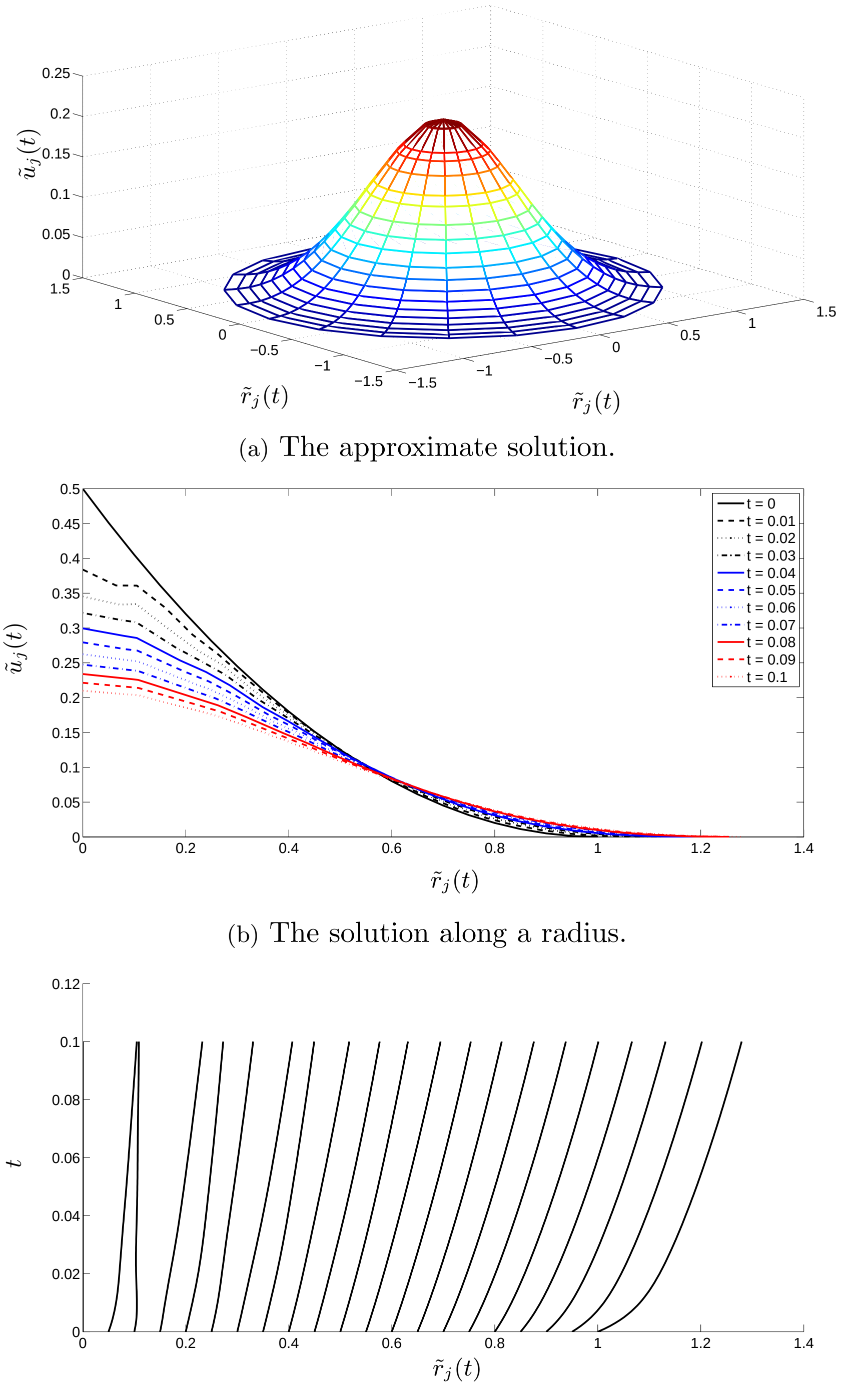


(a) The approximate solution.

(b) The solution along a radius.

(c) The mesh trajectory where $\tilde{r}_{19}(t)$ is red, and $\tilde{r}_{20}(t)$ is blue.

Fig. 6.6: The radial Crank-Gupta problem using equation (6.44) to calculate the boundary position, $N = 20$, $\Delta t = 1 \times 10^{-4}$.



(a) The approximate solution.

(b) The solution along a radius.

(c) The mesh trajectory.

Fig. 6.7: The radial Crank-Gupta problem using extrapolation for $\tilde{s}_N(t)$, $N = 20$, $\Delta t = 1.25 \times 10^{-4}$.

solve for $t \in [0, 0.1]$ and compute results for $N = 10 \times 2^{\hat{N}-1}$, $\hat{N} = 0, \dots, 5$. In order to compare results for different values of $\hat{N}$, we denote the points of the mesh for a particular value of $\hat{N}$ by $x_{j,\hat{N}}$, $j = 0, \dots, (10 \times 2^{\hat{N}-1})$. We then compute both $x_{2^{\hat{N}-1}i,\hat{N}}$ and $u_{2^{\hat{N}-1}i,\hat{N}} \approx u(x_{2^{\hat{N}-1}i,\hat{N}}, 5)$ for each $i = 0, \dots, 10$ as $\hat{N}$ increases. We compare the numerical outcomes with the exact solution from (6.47), at $t = 0.1$,

$$\bar{u}_{2^{\hat{N}-1}i,\hat{N}} = e^{x_{2^{\hat{N}-1}i,\hat{N}}-0.9} - x_{2^{\hat{N}-1}i,\hat{N}} - 0.1,$$

where $\bar{u}_{2^{\hat{N}-1}i,\hat{N}}$ is the exact solution at the calculated mesh points.

To balance the spatial and temporal errors, and recalling that we have used explicit Euler time-stepping, we use $\Delta t = \mathcal{O}\left(\frac{1}{N^2}\right)$, precisely $\Delta t = \frac{1}{200(4^{\hat{N}})}$. We anticipate that the pointwise errors $|\bar{u}_{2^{\hat{N}-1}i,\hat{N}} - u_{2^{\hat{N}-1}i,\hat{N}}|$ will decrease as $\hat{N}$ increases, for each $i = 0, \dots, 10$.

As a measure of the errors, we calculate

$$E_N(u) = \sqrt{\frac{\sum_{i=0}^{10}(\bar{u}_{2^{\hat{N}-1}i,\hat{N}} - u_{2^{\hat{N}-1}i,\hat{N}})^2}{\sum_{i=0}^{10}(\bar{u}_{2^{\hat{N}-1}i,\hat{N}})^2}},$$

for $\hat{N} = 0, \dots, 5$ (i.e. $N = 10, 20, 40, 80, 160, 320$). We investigate the hypothesis that

$$E_N(u) \sim \frac{1}{N^p}, \tag{6.65}$$

for large N, where p is the estimated order of convergence. If (6.65) holds then we would expect that p_{2N} defined by

$$p_{2N} = -\log_2\left(\frac{E_{2N}(u)}{E_N(u)}\right),$$

would approach the constant values p and q as $N \to \infty$. Since each step of our scheme is second order in space and first order in time, and recalling that $\Delta t = \mathcal{O}\left(\frac{1}{N^2}\right)$, we might expect to see $p \approx 2$. Convergence results are shown in Table 6.3. We see that $E_N(u)$

N	$E_N(u)$	p_{2N}
10	7.581×10^{-3}	-
20	2.502×10^{-3}	1.6
40	6.796×10^{-4}	1.9
80	1.825×10^{-4}	1.9
160	4.879×10^{-5}	1.9
320	1.235×10^{-5}	2.0

Table 6.3: Relative errors for u with rates of convergence using the explicit Euler time-stepping scheme.

decreases as N increases for each of the moving mesh methods. This strongly suggests that as the number of nodes increases, the solution $\tilde{u}_j(t)$ converges. The p-values presented strongly imply second-order convergence of the solution $\tilde{u}_j(t)$. These results, together with the comparisons with the Fourier Series approach, suggest that our moving mesh method is accurate, and has second order convergence.

Figures 6.8(a)–6.8(c) show the results from imposing the alternative boundary condition. The solution to the original problem is very small for $t = 0.19$, see Figure 6.3(a), whereas the modified problem takes much longer to decrease. This is partly because the outer boundary moves in at a slower rate for the modified problem, which can be seen by comparing Figures 6.3(b) and 6.8(b) (where we observe that the boundary moves in linearly). Lastly, from Figure 6.8(c) we note that the nodes move in a fairly uniform manner, remaining reasonably equidistant.

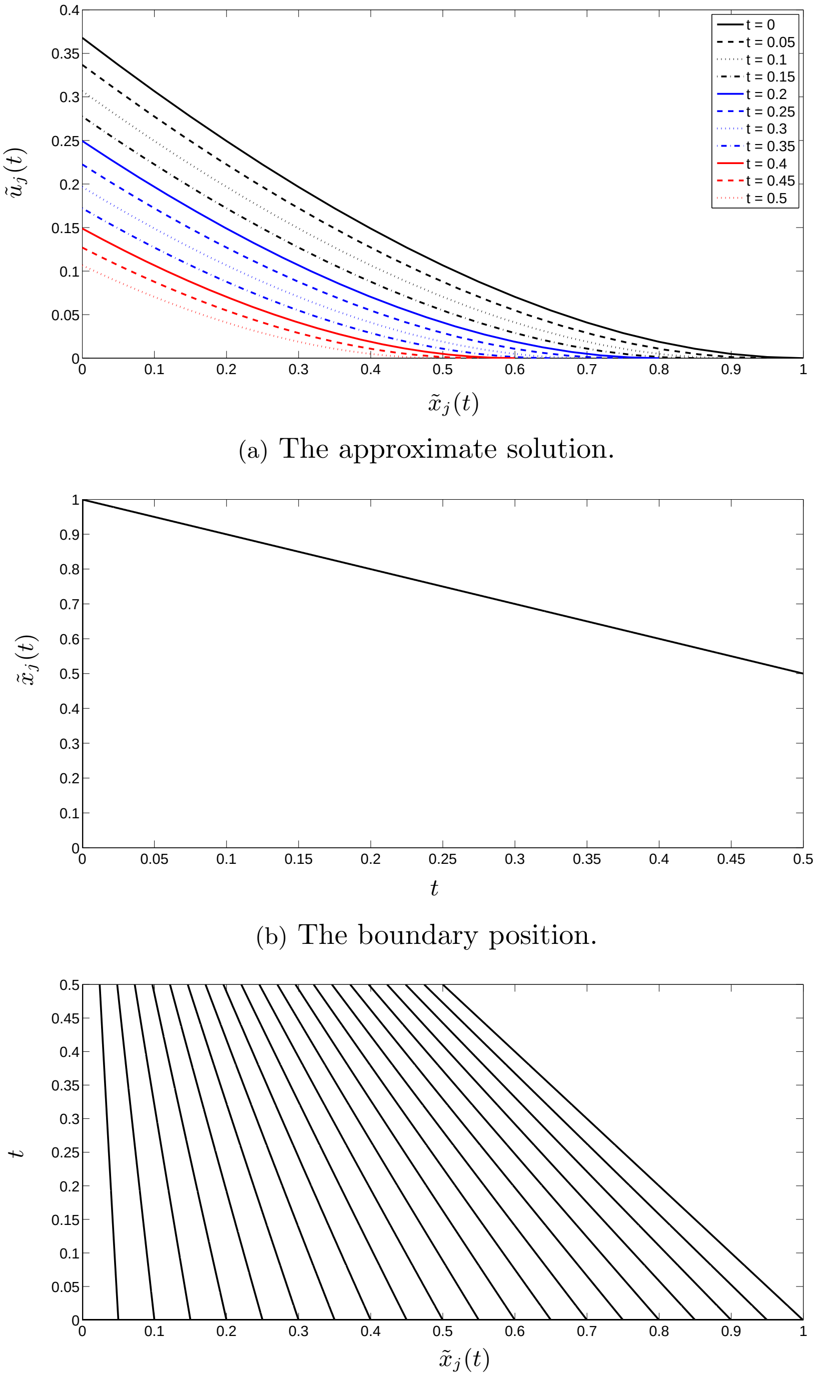
Figures 6.9(a) and 6.9(b) compare the numerical solution with the exact solution. Figure 6.9(a) shows the differences between the exact solution and numerical solution over the region at $t = 0.5$. We note that the difference is of order 10^{-5}, with the largest difference at the inner boundary. Bearing this in mind, Figure 6.9(b) shows the difference between the exact and numerical solution at the inner boundary. Interestingly, we note that the relative error increases until about $t = 0.25$, and then decreases again.

6.7.4 A partial mass balance moving mesh method

The Crank-Gupta problem was solved using the moving mesh method which preserves partial mass balances. We used the second order approximations (6.61) to compute the mesh velocity, and the first-order approximation (3.34) to compute the updated solution. We also used Laplacian smoothing on the updated solution at each time-level.

When using (6.34) to calculate the boundary position, the mesh tangles, see Figure 6.10(a). Figures 6.10(b) and 6.10(c) use a polynomial extrapolation for the mesh velocity $\tilde{x}_{10}(t)$. We notice that the boundary moves out very rapidly. This spread increases out of control very quickly, as these figures show the results for very small times. We conclude that balancing the rate of change of partial masses with the source term is an improper approach for numerically solving the Crank-Gupta problem. This may be because the mesh velocity from the approach (6.59) does not explicitly depend upon the boundary position. It is more appropriate to not separate the two terms of the right-hand side of (6.1), as in §6.3.

We have completed our application of our finite difference moving mesh method to the Crank-Gupta problem. In the next section we present our results from applying the finite element moving mesh method of Baines, Hubbard and Jimack [5] to the Crank-Gupta problem.

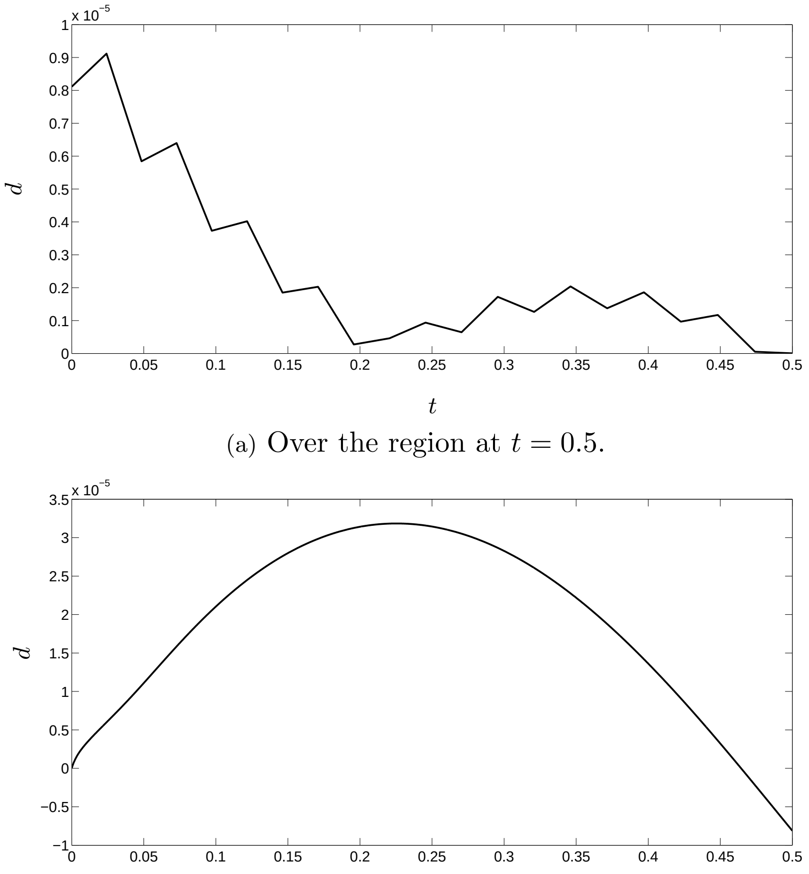


(a) The approximate solution.

(b) The boundary position.

(c) The mesh trajectory.

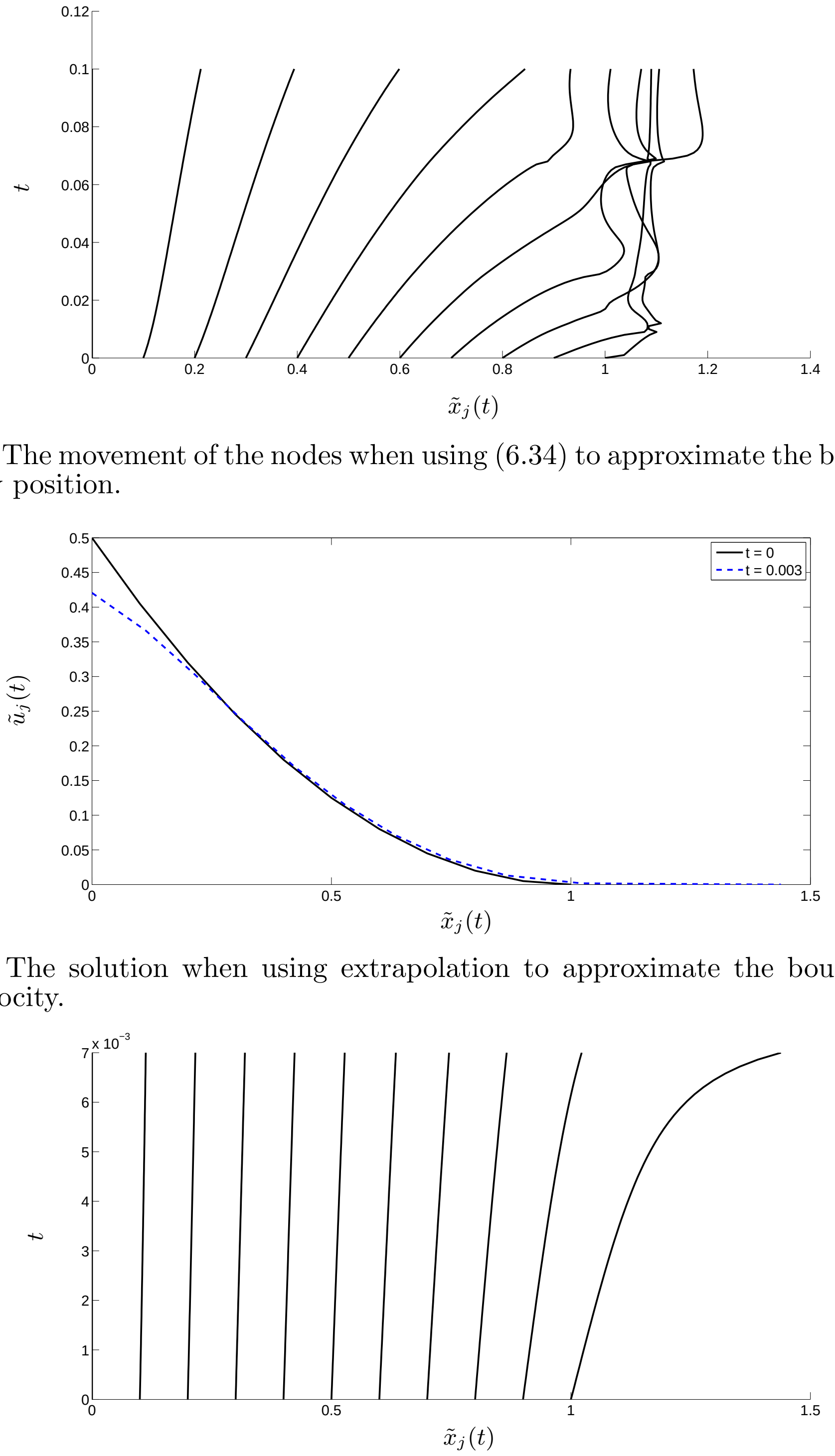
Fig. 6.8: The Crank-Gupta PDE with alternative boundary conditions solved using conservation of partial masses, $N = 20$, $\Delta t = 2 \times 10^{-4}$.



(a) Over the region at $t = 0.5$.

(b) Over time at $j = 0$.

Fig. 6.9: The difference between the exact solution and numerical solution.



(a) The movement of the nodes when using (6.34) to approximate the boundary position.

(b) The solution when using extrapolation to approximate the boundary velocity.

(c) The movement of the nodes when using extrapolation to approximate the boundary velocity.

Fig. 6.10: The Crank-Gupta problem solved using the mass balance method, $N = 10$, $\Delta t = 1 \times 10^{-5}$.

6.8 Finite element method

As mentioned in §2.3, the moving mesh method we use is a finite difference version of the Conservation Method given in [5]. We also solved the Crank-Gupta problem using the Conservation Method with finite elements. The numerical process is given in §2.3 (with an additional calculation for updating the mass at each time-level), but the details are omitted since details from solving this problem are given in [5]. However, unlike [5], we have used Matlab and reconstructed the mesh at each time step using a Delaunay Triangulation as described in §4.8.

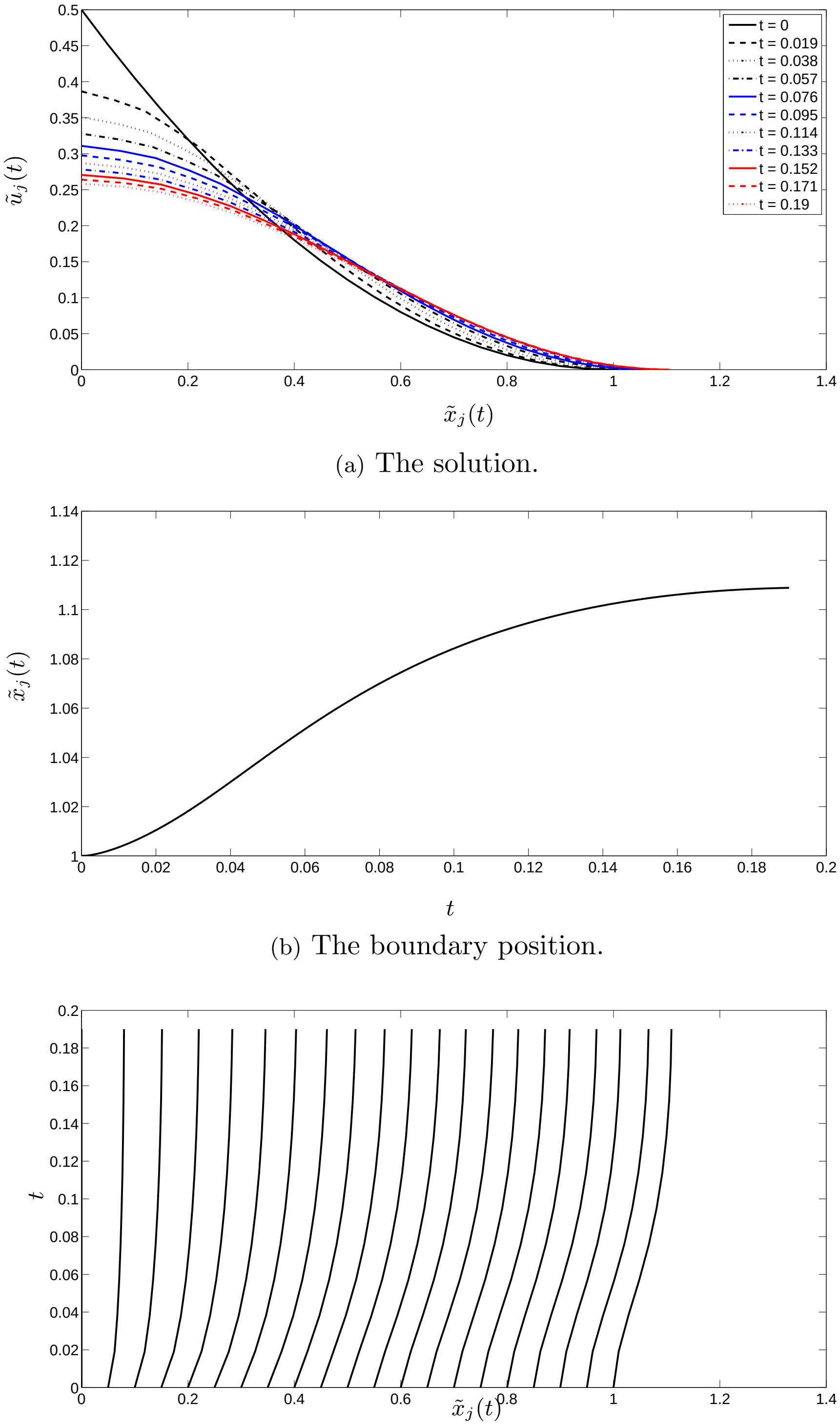
We see in the one-dimensional case, Figures 6.11(a)–6.11(c) that the boundary moves out, when it should be moving in. This inaccuracy may be from one-sided approximations taken at the boundary. In the two-dimensional case, Figure 6.12(a)–6.12(b), the outer boundary is moving in, as desired, but the inner boundary condition of $u_x = 0$ is not satisfied. This poor representation at the centre may have occured since we used a radial mesh. The Crank-Gupta numerical solution in [5] do not present these boundary inaccuracies, so we conclude that we have not fully implemented their method.

In the next section we summarise our work on the Crank-Gupta problem.

6.9 Summary for the Crank-Gupta problem

The Crank-Gupta problem was derived to model the diffusion of oxygen in absorbing tissue. In this chapter we applied a moving mesh numerical method to solve the Crank-Gupta problem originally presented in [38]. Applying our method to this problem is an advance from the PME and Richards' equation since the solution has a decreasing mass, so we needed the additional calculation of updating the total mass at each time-level.

When determining a self-similar solution for the Crank-Gupta problem, we observed that deriving an analytical solution which satisfies the boundary conditions is not straightforward. This provides motivation for a numerical approach. We used conservation of partial masses, and the mass balance method, with explicit time-stepping. We did not give details about implementing a semi-implicit time-stepping since we found that using the semi-implicit scheme given in §3.4 does not allow the total mass and the mesh to be updated simultaneously. This can cause errors as we showed in the results section. Since we were unable to obtain a series solution that satisfied the boundary conditions, in the results section we compared the original Crank-Gupta problem to a Fourier Series solution from [39]. This only allows us to compare a single value of the solution, so we also solved the Crank-Gupta problem with a different inner boundary condition, from [5], which has

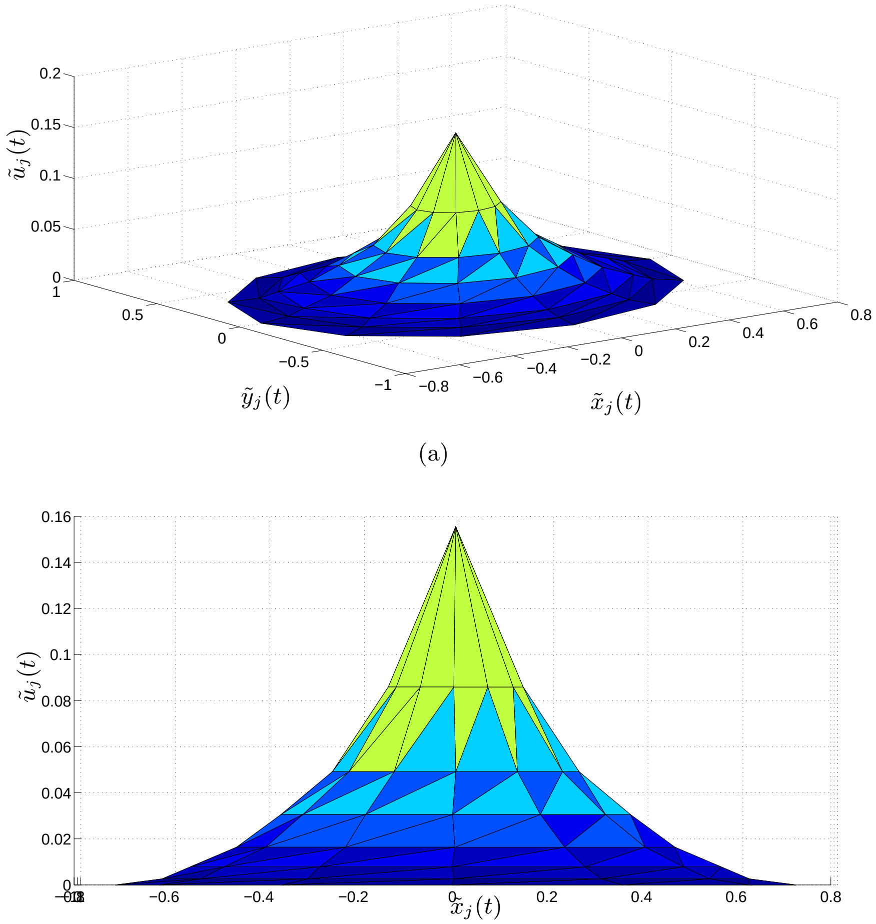


(a) The solution.

(b) The boundary position.

(c) The movement of the nodes.

Fig. 6.11: The one-dimensional Crank-Gupta PDE solved using finite elements, $N = 20$, $\Delta t = 1 \times 10^{-4}$.



(a)

(b)

Fig. 6.12: The two-dimensional Crank-Gupta PDE solved using finite elements, $\Delta t = 1 \times 10^{-4}$, final time $t = 0.1$.

an exact solution. Both comparisons indicated convergence and accuracy of our moving mesh method that conserves partial masses. It appears that the mesh and solution have second-order convergence. There was no analysis of the method that balances the partial masses with the source term since we found that the boundary moves out very rapidly, which is highly inaccurate, so we conclude that it is not a suitable method to solve the Crank-Gupta problem numerically. We also came across inaccuracies at the outer boundary when numerically solving the two-dimensional radially symmetric case. We attempted several different ways to capture the boundary behaviour, but the transformation to the radially symmetric case appears to be more challenging than with the PME, since we did not achieve accurate results.

We used the Conservation Method of Baines, Hubbard and Jimack [5] to solve the Crank-Gupta problem in one and two dimensions. We did not give much detail since this problem is discussed in [5]. We find the solutions mostly exhibit the desired behaviour.

Having considered two problems which conserve mass, and one problem that decreases in mass, we now consider the more complicated problem of modelling tumour growth, which involves a system of equations. We begin by giving a background on cancer research, and a brief history of mathematics and tumour modelling.

7
A Tumour Growth Problem

7.1 A brief background on cancer growth

A tumour is a group of cancer cells. Like all cells, these cells gain nutrients, such as glucose and oxygen, from the surrounding environment. There is disagreement as to what causes a normal cell to turn into a cancer cell. Nonetheless, it is generally accepted that once a tumour is initiated it has three successive growth stages that it can possibly go through.

The first stage is referred to as the avascular stage. At this early stage the tumour has a large surface area in relation to its size. Consequently, a large proportion of the cells benefit from the surrounding nutrients and proliferate, causing the tumour to grow rapidly. This can only continue to a certain size. As the tumour grows the external nutrients cannot diffuse into the cells in the centre. The cells in the centre enter into a quiescent state. In this state, they are dormant - but are able to proliferate again should nutrients become available to them. As the cells on the edge continue to proliferate, i.e. the tumour grows, the proliferating region expands and the cells in the centre die creating a necrotic core. At this stage, there is a balance between the maximum possible size of the tumour and its surrounding environment, the key reason being the limited ability for the majority of cells to obtain nutrients. However, some mathematical models include other relevant factors such as surface tension [81], [63], attractive cell forces [22], residual stress [2] and contractility (possibly due to the wound-healing process) [69].

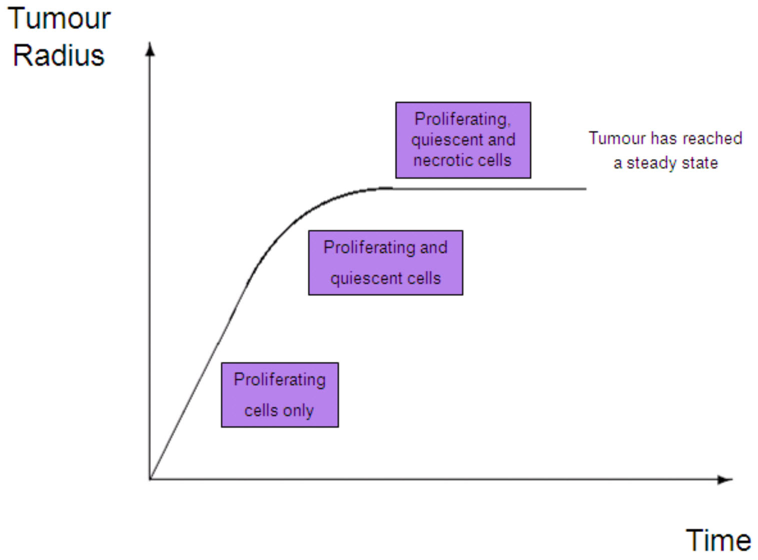

Tumour Radius
Proliferating, quiescent and necrotic cells
Tumour has reached a steady state
Proliferating and quiescent cells
Proliferating cells only
Time

Fig. 7.1: The growth of an avascular tumour.

During the avascular stage, tumours are benign and are unlikely to affect the host. However, once the tumour obtains a dependable blood supply from a nearby capillary, then it advances to the more aggressive vascular stage. A nutrient rich capillary is drawn into the tumour, initiating the rapid growth of cancerous cells. The process by which the tumour obtains its own blood supply is called angiogenesis and preventing this from occurring is of particular interest to drug development. This is because once the tumour has obtained a blood supply the tumour can leave its primary location via the circulatory system (metastasis) and settle in multiple areas of the body. The metastatic stage is the final stage of tumour growth, and the most difficult to treat.

These three distinct stages have different characteristics so require individual investigation. We shall study the primary stage, avascular tumour growth.

7.2 Avascular research

As previously mentioned, the later stages of tumour growth are more critical since it is usually not until after angiogenesis that cancer is detrimental to the hosts' health. During the avascular stage, the tumour is benign. Indeed, following a study of human cancers in mice [98] there is a recent controversial hypothesis that we all have small dormant avascular tumours in our bodies.

Regardless of this clinical viewpoint, avascular tumour growth warrants the interest of scientists. It is beneficial to understand the simple system and its components prior to attempting analysis of a more complex system. Vascular tumours have many of the same characteristics as avascular tumours, but the quantity and quality of data on avascular tumours is of a higher standard. This is because it is comparatively easier and cheaper to reproduce high quality avascular tumour experimental evidence in *in vitro* form.

In summary, we will be investigating a model for avascular tumours (see Figure 7.2) as they are simpler to model and help give an insight into the mechanisms of vascular tumour growth.

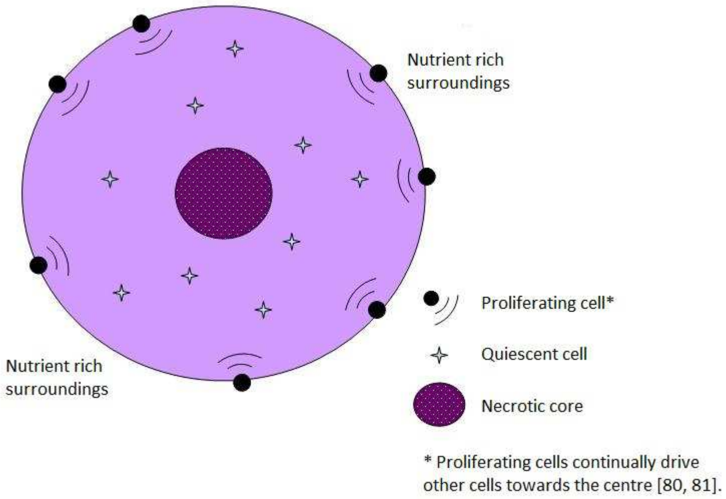

Nutrient rich surroundings
Nutrient rich surroundings
Proliferating cell*
Quiescent cell
Necrotic core
* Proliferating cells continually drive other cells towards the centre [80, 81].

Fig. 7.2: An avascular tumour.

7.3 The role of mathematics in cancer research

Ever since complex life evolved, it has been susceptible to cancer. The oldest description of cancer in humans was found in an Egyptian papyrus written between 3000-1500 BC. Today specialists are still extensively researching and experimenting in attempts to find cures and improve treatments. Cancer is rife in modern society - and thereby the size of the cancer research industry is vast.

Despite the considerable volumes of time and money that is invested in this industry, the tools of mathematics have not really been exploited. The UK's first specialist cancer research organisation was set up in 1902, yet mathematics only started making a major contribution in this field from the early seventies. Most of the research is in molecular biology, cell biology and drug delivery. However, the use of mathematics to aid cancer research is increasing by way of computational modelling, as well as analysis on the large library of experimental data.

Indeed, it has been noted that a conceptual framework within which all these new (and old) data can be fitted is lacking [49]. In [47] it states that 'clinical oncologists and tumour biologists posses virtually no comprehensive theoretical model to serve as a framework for understanding, organising and applying these data'. By being educated as to which mechanisms are critical to the essence of tumour growth, these could possibly be manipulated to our advantage. As Byrne [30] remarks, 'In order to gain such insight, it is usually necessary to perform large numbers of time-consuming and intricate experiments - but not always. Through the development and solution of mathematical models that describe different aspects of solid tumour growth, applied mathematics has the potential to prevent excessive experimentation'.

Ideally, experiments and modelling work hand-in-hand. The experiments can not only prove to be costly, but the subtleties of the many intricate processes can easily be overlooked. By modelling tumour growth to mimic data already collected, potentially pivotal characteristics can be identified. This can ensure that these interactions are monitored closely in future experiments. Ultimately, the aim for applied mathematics in tumour growth is to enlighten biologists as to the key processes, so that these can be artificially altered in a manner that eradicates (or manages) the disease.

Parameterisation

Gaining parameters for tumour growth is a challenge within itself. There are many variables on varying scales - some of which can be difficult to measure. For example, the *in vivo* measurement of a pressure that is probably very low ($\sim$ 10 mmHg) in a sample that is very small in size (max 1mm) is technically very difficult [84]. 'An important role of modelling in this respect is to determine, via sensitivity and/or bifurcation analysis, on which parameters the behaviour of the model crucially depends, thereby identifying which parameters need to be measured correctly' [84]. When choosing parameters, the values may be chosen to show qualitative predictions. An example is given in [46] where a phenomenon was shown to be largely independent of the specific parameter values. All the same, if parameters are available then a well-parameterised model can make quantative *and* qualitative predictions.

7.4 A mathematical model of tumour growth

Mathematical models of tumour growth can offer effective and efficient ways to advance our understanding of cancer research; see, for example, the survey papers [2, 84]. In recent years there has been a large increase in the number of PDE models describing solid tumour growth.

Whilst differences between such models exist, many exhibit the following features:

- Equations describing the diffusion of nutrients or growth factors in and around the tumour region (generally parabolic in type);

- Mass transfer equations describing the dynamic variation in tumour tissue (generally hyperbolic);

- Mass balance equations describing the growth of the tumour (generally elliptic).

All of these equations are generally coupled via nonlinear interactions. For instance, the growth dynamics of a specific cell type may depend in a nonlinear way on a specific nutrient or growth factor. Examples include Ward and King [103] who developed a two phase model of a growing multicellular tumour spheroid (MCTS) in which cells were considered to exist in either a live or dead state, whilst Please *et al.* [80, 81] considered the two phases to be live cells and water, respectively. In contrast Tindall and Please [96] considered a three phase model to account for proliferating and quiescent cells and dead cell material. The complexity of such nonlinear mathematical models means they are most often solved and investigated numerically. Given the coupling between the various equation types (parabolic, hyperbolic and elliptic) it is important that such methods are robust and accurate.

In this work we consider a recent two phase model of tumour growth developed by Breward *et al.* [22], which is a specific form of the two-phase model in [29]. The two phases, cell and water, each have an associated velocity, pressure and volume-fraction-averaged stress tensor. We utilise the model to compare a number of moving mesh strategies with the commonly employed fixed numerical mesh approach. Although three phase models may incorporate more detail our aim here is to demonstrate that moving mesh methods are an effective tool for the numerical solution of problems such as tumour growth models, and for this purpose a two phase model suffices. The extension to three phase models is technical but straightforward in principle. We focus on a two-phase model to demonstrate clearly the velocity-based moving mesh schemes, which can be adapted to numerically solve more sophisticated models.

Our first task is to non-dimensionalise the model. This involves the partial or full removal of units by a suitable substitution of variables. Non-dimensionalisation can simplify a problem by reducing the number of variables. It also aids analysis of the behaviour of a system by recovering characteristic properties. In our case, the key motivator to non-dimensionalising the system is to enable us to take advantage of parameterisations studied elsewhere.

In the next subsection we present the normalised one-dimensional model proposed in [22], followed by §7.5 where we surmise the fixed numerical mesh method used in [22], so as to compare results with our moving mesh strategies, of which there are three. The details of these strategies are given in §7.6, where we solve the tumour growth model numerically using each one in turn. The results from the fixed mesh method and the three moving mesh methods are discussed in §7.7. Finally, in §7.8 we conclude that a moving mesh method can prove to be an elegant and accurate numerical approach that updates the mesh smoothly with the solution of the orginal model, whilst preserving chosen features of the model such as local mass balance, or relative partial masses. However, since the mesh depends upon the model, care must be taken when choosing a feature of the model to preserve.

Model formulation

The model assumes the tumour consists of two phases, water and live cells, which are treated as incompressible fluids whose densities are equal, to leading order. The model is derived by applying mass balance to the cell and water phases. Further assumptions made are that inertial effects are negligible, no external forces act on the system, and, on the timescale of interest, the cell and water phases can be treated as viscous and inviscid fluids respectively. The model is applied to a tumour whose growth is parallel to the x-axis, and is symmetric about its midpoint. We have altered the notation to be consistent with previous chapters.

From [22] the non-dimensional model, in Cartesian form, for the volume fraction of cells $u(x,t) \in (0,1)$, with $t > 0$ and $x \in [0, b(t)]$, where $b(t)$ is the tumour radius, is comprised of

$$\begin{aligned}
\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(wu) &= \frac{(1+s_1)u(1-u)C}{1+s_1C} - \frac{s_2+s_3C}{1+s_4C}u =: S(u,C), && (7.1)\\
\frac{\partial}{\partial x}\left[\mu u \frac{\partial w}{\partial x} - u\frac{u-u^*}{(1-u)^2}\mathrm{H}(u-u_{\min})\right] &= \frac{kuw}{1-u}, && (7.2)\\
\frac{\partial^2 C}{\partial x^2} &= \frac{QuC}{1+Q_1C}, && (7.3)
\end{aligned}$$

where $w(x,t)$ is the cell velocity, $C(x,t)$ is the nutrient concentration and H is the Heaviside function. The volume fraction of water is $1-u$. The first term of $S(u,C)$ in (7.1) represents cell growth due to mitosis (cell division), and the second term represents cell death. The parameters μ (a combination of the shear and bulk viscosities), k (the drag coefficient), and s_1, s_2, s_3, s_4, Q and Q_1 are all positive constants. In addition, $u_{\min}$ and u^* (a natural cell packing density) are constants such that $0 < u_{\min} < u^* < 1$. We remark that equation (7.1) arises from the global mass balance equation,

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_0^{b(t)} u(x,t)\,\mathrm{d}x = \int_0^{b(t)} S(u,C)\,\mathrm{d}x. \qquad (7.4)$$

The normalised model has initial and boundary conditions

$$\begin{aligned} b = 1,\ \ u = u^0(x) &\quad \text{at} \quad t = 0, \qquad (7.5)\\ w = \frac{\partial C}{\partial x} = 0 &\quad \text{at} \quad x = 0,\ t > 0, \qquad (7.6)\\ \mu\frac{\partial w}{\partial x} - \frac{u-u^*}{(1-u)^2}\mathrm{H}(u-u_{\min}) = 0,\ \ C = 1,\ \ \frac{\partial b}{\partial t} = w &\quad \text{at} \quad x = b,\ t > 0. \qquad (7.7) \end{aligned}$$

Remark 7.4.1 *We observe that for the case of zero viscosity $\mu = 0$, equations (7.1) and (7.2) reduce to*

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left\{\frac{1-u}{k}\frac{\partial}{\partial x}\chi(u)\right\} + S(u, C),$$

where $\chi(u) = u\frac{u-u^*}{(1-u)^2}\mathrm{H}(u-u_{\min})$, so the PDE is now of the form (3.27).

In the next three subsections we show that moving the mesh to preserve features of the model can produce results in line with [22]. We also present results which demonstrate that the local feature of the model used to track the nodes needs to be carefully chosen. However we first consider the numerical approach that is usually employed: rescaling the problem to a fixed mesh.

7.5 Rescaling to a fixed numerical mesh

In [22] the moving domain $x \in [0, b(t)]$ is mapped to a fixed numerical domain $\xi \in [0, 1]$ by the transformation $\xi = \frac{x}{b(t)}$, $\tau = t$. This is a common approach for problems of this kind since it allows standard numerical methods to be applied. However, there is a danger of disturbing the original balance laws when using this technique.

Using the chain rule to differentiate $u(\xi, \tau)$ with respect to time τ, the transformed problem is

$$\begin{aligned} \frac{\partial u}{\partial \tau} - \frac{\xi}{b}\frac{\mathrm{d}b}{\mathrm{d}\tau}\frac{\partial u}{\partial \xi} + \frac{1}{b}\frac{\partial}{\partial \xi}(wu) &= S(u, C), \qquad (7.8)\\ \frac{\partial}{\partial \xi}\left(\mu u\frac{\partial w}{\partial \xi} - bu\frac{u-u^*}{(1-u)^2}\mathrm{H}(u-u_{\min})\right) &= \frac{kb^2uw}{1-u}, \qquad (7.9)\\ \frac{\partial^2 C}{\partial \xi^2} &= \frac{Qb^2uC}{1+Q_1C}, \qquad (7.10) \end{aligned}$$

with initial and boundary conditions

$$
\begin{aligned}
b = 1,\;\; u = u^0(x) \quad &\text{at} \quad \tau = 0, &(7.11)\\
w \;=\; \frac{\partial C}{\partial \xi} \;=\; 0 \quad &\text{at} \quad \xi = 0,\; \tau > 0, &(7.12)\\
\mu\frac{\partial w}{\partial \xi} - b\frac{u - u^*}{(1-u)^2}\mathrm{H}(u - u_{\min}) = 0,\;\; C = 1,\;\; \frac{\mathrm{d}b}{\mathrm{d}\tau} = w \quad &\text{at} \quad \xi = 1,\; \tau > 0. &(7.13)
\end{aligned}
$$

We note that in this approach the spatial derivatives in equation (7.8), unlike in the original equation (7.1), are not in divergence form, i.e. there is an additional term on the left-hand side that is not a total derivative with respect to ξ. This changes the structure of the equation which can lead to an inappropriate numerical approximation.

Although details of the numerical method are not given in [22], in order to compare our results to those in [22] we surmise their numerical method to produce similar results. Many authors utilise the National Algorithms Group (NAG) routine D02RAF, which uses a finite difference approach [103]. Using the above equations we postulate an algorithm in which we choose a time step $\Delta\tau > 0$ and divide the region $(0, 1)$ into N equal cells of size $\Delta\xi = \frac{1}{N}$. We define $\xi_j = j\Delta\xi$, $j = 0, 1, \ldots, N$, and $\tau^m = m\Delta\tau$, $m = 0, 1, \ldots$, and approximations $u_j^m \approx u(\xi_j, \tau^m)$, $b^m \approx b(\tau^m)$, $w_j^m \approx w(\xi_j, \tau^m)$, $C_j^n \approx C(\xi_j, \tau^m)$, and $S_j^m \approx S(u(\xi_j, \tau^m), C(\xi_j, \tau^m))$. Given u_j^m, we compute C_j^m, w_j^m and ultimately u_j^{m+1} by a series of steps (labelled Steps F1–F4 below):

Step F1: Find C_j^m by applying central finite differences to (7.10),

$$
\frac{C_{j-1}^m - 2C_j^m + C_{j+1}^m}{(\Delta\xi)^2} = \frac{Q(b^m)^2 u_j^m C_j^m}{1 + Q_1 C_j^m}, \tag{7.14}
$$

for $j = 0, 1, \ldots, N-1$, where from (7.12) and (7.13), we take $C_{-1}^m = C_1^m$ and $C_b^m = 1$. Newton's method is used to solve the subsequent system of nonlinear equations when $Q_1 \neq 0$.

Step F2: Find w_j^m by applying central finite differences to (7.9),

$$
\begin{aligned}
\frac{1}{\Delta\xi}\Bigg\{ u_{j+\frac{1}{2}}^m &\left[\mu\frac{w_{j+1}^m - w_j^m}{\Delta\xi} - b^m\frac{u_{j+\frac{1}{2}}^m - u^*}{(1 - u_{j+\frac{1}{2}}^m)^2}\mathrm{H}(u_{j+\frac{1}{2}}^m - u_{\min})\right]\\
-u_{j-\frac{1}{2}}^m &\left[\mu\frac{w_j^m - w_{j-1}^m}{\Delta\xi} - b^m\frac{u_{j-\frac{1}{2}}^m - u^*}{(1 - u_{j-\frac{1}{2}}^m)^2}\mathrm{H}(u_{j-\frac{1}{2}}^m - u_{\min})\right]\Bigg\} = \frac{k(b^m)^2 u_j^m}{1 - u_j^m} w_j^m,
\end{aligned} \tag{7.15}
$$

for $j = 1, 2, \ldots, N-1$, where $u_{j+\frac{1}{2}}^m = \frac{1}{2}(u_j^m + u_{j+1}^m)$ and $u_{j-\frac{1}{2}}^m = \frac{1}{2}(u_{j-1}^m + u_j^m)$, leading to a linear system of equations. At the inner boundary $w_0^m = 0$, as given by (7.12).

To determine u_N^m, we discretise the boundary condition (7.13) by taking values $(\cdot)_{N-\frac{1}{2}}^m$ and $(\cdot)_{N+\frac{1}{2}}^m$ (the average about $(\cdot)_N^m$) to obtain

$$\begin{aligned} &\frac{1}{2}\left[\mu\frac{w_{N+1}^m - u_N^m}{\Delta\xi} - b^m\frac{u_{N+\frac{1}{2}}^m - u^*}{(1-u_{N+\frac{1}{2}}^m)^2}\mathrm{H}(u_{N+\frac{1}{2}}^m - u_{\min})\right] \\ - &\frac{1}{2}\left[\mu\frac{w_N^m - w_{N-1}^m}{\Delta\xi} - b^m\frac{u_{N-\frac{1}{2}}^m - u^*}{(1-u_{N-\frac{1}{2}}^m)^2}\mathrm{H}(u_{N-\frac{1}{2}}^m - u_{\min})\right] = 0. \qquad (7.16)\end{aligned}$$

We then adapt (7.15) for $j = N$, using (7.16) to replace the first term in square brackets, leading to

$$-\frac{u_{N+\frac{1}{2}}^m + u_{N-\frac{1}{2}}^m}{\Delta\xi}\left[\mu\frac{w_N^m - w_{N-1}^m}{\Delta\xi} - b^m\frac{u_{N-\frac{1}{2}}^m - u^*}{(1-u_{N-\frac{1}{2}}^m)^2}\mathrm{H}(u_{N-\frac{1}{2}}^m - u_{\min})\right] = \frac{k(b^m)^2u_N^m}{1-u_N^m}w_N^m,$$

where $u_{N+\frac{1}{2}}^m + u_{N-\frac{1}{2}}^m = 2u_N^m$. This yields a complete set of linear equations for the velocity w_j^m, $j = 1, \ldots, N$.

Step F3: Discretise (7.8) using an explicit Euler time-stepping scheme and a central difference approximation in space, giving

$$\frac{u_j^{m+1} - u_j^m}{\Delta t} = \frac{jw_N^m(u_{j+1}^m - u_{j-1}^m)}{2b^m} - \frac{w_{j+1}^m u_{j+1}^m - w_{j-1}^m u_{j-1}^m}{2b^m\Delta\xi} + S_j^m,$$

for $j = 1, 2, \ldots, N-1$. One-sided approximations are used at the boundaries.

Step F4: Since the tumour radius moves with the cell velocity at the boundary, we calculate the tumour radius at the new time-level using

$$b^{m+1} = b^m + \Delta t w_N^m.$$

We then return to Step F1 to complete the next time step. This numerical scheme produces results in line with those in [22] (see §7.7). Although this is a perfectly reasonable scheme, in the next section we compare it with moving mesh methods by solving the same problem numerically. We use a velocity-based moving mesh approach in which the velocities are defined by three different strategies.

7.6 Moving mesh methods

The key component of a velocity-based moving mesh method is the criterion used to define the mesh velocity. We investigate three different choices here, in which we move the mesh in the following ways:

Method A - proportional to the boundary position $b(t)$. This construction is geometrical in nature and is very similar to the method described above;

Method B - proportional to the local cell velocity w, i.e. based on a feature which is observed over the whole tumour. We discover that this method preserves partial mass balances, making it the same as the method described in §3.3;

Method C - in such a way as to conserve local mass fractions of the solution u in time (the method described in §3.2). Like Method B, this is based on a prevalent feature of the model.

For all of these moving mesh methods (and in contrast to some fixed mesh methods), the final mesh node tracks the tumour radius.

In the model given by equations (7.1)–(7.7), x is an independent variable. We introduce the dependent variable $\tilde{x}_j(t)$, $j = 0, \ldots, N$, to represent the $N+1$ nodes of the mesh, which are dependent on t. The mesh is initially equally-spaced; however, unlike the fixed mesh, re-scaling the grid points leads to them becoming irregularly separated, in general. We define the velocity of the j-th node to be

$$v(\tilde{x}_j, t) = \tilde{v}_j(t) = \frac{\mathrm{d}\tilde{x}_j}{\mathrm{d}t}. \tag{7.17}$$

We choose a time step $\Delta t > 0$ and define $t^m = m\Delta t$, $m = 0, 1, \ldots$. We denote $\tilde{x}_j(t^m)$ by x_j^m, and use the approximations $u_j^m \approx \tilde{u}_j(t^m)$, $w_j^m \approx \tilde{w}_j(t^m)$, $C_j^m \approx \tilde{C}_j(t^m)$, and $v_j^m \approx \tilde{v}_j(t^m)$. For a given x_j^m and u_j^m, $j = 0, \ldots, N$, we compute C_j^m, w_j^m, v_j^m, x_j^{m+1} and u_j^{m+1} by the following algorithm:

Step 1: Find C_j^m by approximating (7.3) (with boundary conditions given by (7.6) and (7.7)) using central finite differences on the non-uniform mesh $\{x_0^m, \cdots, x_N^m\}$. The resulting set of equations is similar to (7.14), of the form

$$T^l C_{j-1}^m + T^d C_j^m + T^u C_{j+1}^m \quad = \quad \frac{Q u_j^m C_j^m}{1 + Q_1 C_j^m}, \qquad j = 0, 1, \ldots, N-1,$$

where

$$\begin{aligned}
T^l &= \frac{2}{(x_j^m - x_{j-1}^m)(x_{j+1}^m - x_{j-1}^m)},\\
T^d &= \frac{-2}{(x_{j+1}^m - x_j^m)(x_j^m - x_{j-1}^m)},\\
T^u &= \frac{2}{(x_{j+1}^m - x_j^m)(x_{j+1}^m - x_{j-1}^m)},
\end{aligned}$$

and where $x_{-1}^m = -x_1^m$, $C_{-1}^m = C_1^m$ and $C_N^m = 1$, from the boundary conditions (7.6) and (7.7).

Step 2: Find w_j^m by applying central finite differences to (7.2) on the non-uniform mesh $\{x_0^m, \ldots, x_N^m\}$ with boundary conditions given by (7.6) and (7.7). The resulting set of equations is similar to (7.15) and takes the form

$$\begin{aligned}
\frac{1}{x_{j+\frac{1}{2}}^m - x_{j-\frac{1}{2}}^m} \Bigg\{ u_{j+\frac{1}{2}}^m \left[\mu \frac{w_{j+1}^m - w_j^m}{x_{j+1}^m - x_j^m} - \frac{u_{j+\frac{1}{2}}^m - u^*}{(1 - u_{j+\frac{1}{2}}^m)^2} \mathrm{H}(u_{j+\frac{1}{2}}^m - u_{\min}) \right] \\
- u_{j-\frac{1}{2}}^m \left[\mu \frac{w_j^m - w_{j-1}^m}{x_j^m - x_{j-1}^m} - \frac{u_{j-\frac{1}{2}}^m - u^*}{(1 - u_{j-\frac{1}{2}}^m)^2} \mathrm{H}(u_{j-\frac{1}{2}}^m - u_{\min}) \right] \Bigg\} = \frac{k u_j^m}{1 - u_j^m} w_j^m, \quad (7.18)
\end{aligned}$$

where $x_{j+\frac{1}{2}}^m - x_{j-\frac{1}{2}}^m = \frac{1}{2}(x_{j+1}^m - x_{j-1}^m)$, $j = 1, 2, \ldots, N-1$, and $w_0^m = 0$ (from (7.6)). As with the fixed numerical mesh method, to determine the boundary value u_N we discretise the boundary condition (7.7) in a similar way to (7.16) by taking the average at $(\cdot)_{N-\frac{1}{2}}^m$ and $(\cdot)_{N+\frac{1}{2}}^m$, giving

$$\begin{aligned}
&\frac{1}{2} \left[\mu \frac{w_{N+1}^m - w_N^m}{x_{N+1}^m - x_N^m} - \frac{u_{N+\frac{1}{2}}^m - u^*}{(1 - u_{N+\frac{1}{2}}^m)^2} \mathrm{H}(u_{N+\frac{1}{2}}^m - u_{\min}) \right] \\
- &\frac{1}{2} \left[\mu \frac{w_N^m - w_{N-1}^m}{x_N^m - x_{N-1}^m} - \frac{u_{N-\frac{1}{2}}^m - u^*}{(1 - u_{N-\frac{1}{2}}^m)^2} \mathrm{H}(u_{N-\frac{1}{2}}^m - u_{\min}) \right] = 0. \quad (7.19)
\end{aligned}$$

We then adapt (7.18) for $j = N$ using (7.19) to replace the first term in square brackets, leading to

$$-\frac{u_{N+\frac{1}{2}}^m + u_{N-\frac{1}{2}}^m}{x_{N+\frac{1}{2}}^m - x_{N-\frac{1}{2}}^m} \left[\mu \frac{w_N^m - w_{N-1}^m}{x_N^m - x_{N-1}^m} - \frac{u_{N-\frac{1}{2}}^m - u^*}{(1 - u_{N-\frac{1}{2}}^m)^2} \mathrm{H}(u_{N-\frac{1}{2}}^m - u_{\min}) \right] = \frac{k u_N^m}{1 - u_N^m} w_N^m,$$

where $u_{N+\frac{1}{2}}^m + u_{N-\frac{1}{2}}^m = 2u_N^m$. This yields a complete set of linear equations for the velocity w_j^m, $j = 1, \ldots, N$.

Step 3: Calculate the mesh velocity v_j^m. This step will differ for each of Methods A, B and C, and is detailed below.

Step 4: Update the mesh points by the explicit Euler scheme applied to (7.17)

$$x_j^{m+1} = x_j^m + \Delta t v_j^m, \qquad j = 0, 1, \ldots, N,$$

with v_j^m obtained from Step 3.

Step 5: Calculate u_j^{m+1}. The details of this step will again differ for each method used, and are given in §7.6.1, 7.6.2 and 7.6.3 respectively.

When comparing this scheme to the fixed numerical mesh algorithm in §7.5, we see that the first two steps are essentially the same, except with a non-uniform mesh. However, whereas the third step of the algorithm in §7.5 calculates the solution u immediately from (7.8) on the transformed mesh, the moving mesh methods calculate the nodal positions first and then recovers the solution u. Another distinction between the fixed numerical mesh method of §7.5 and the moving mesh methods is that the latter methods preserve a local mass balance through being written in divergence form.

We now give details of each moving mesh method.

7.6.1 Method A

For Method A we move the nodes in Step 3 with a velocity proportional to the velocity of the boundary, i.e.

$$v_j^m = \frac{x_j^m}{x_N^m} w_N^m, \qquad j = 0, 1, \ldots, N.$$

This velocity-based strategy is similar to the numerical mapping in §7.5, see Remark 7.6.1 below. It is geometrical in nature and draws only on information from the boundary of the tumour to determine how to move the nodes. Once the mesh velocity is defined, the new mesh is determined as in Step 4 above.

Now consider Step 5. To recover u on the new mesh we take an integral-based approach. First define the partial masses $\Theta_j(t)$ by

$$\begin{aligned}\Theta_0(t) &= \int_{\tilde{x}_0(t)}^{\tilde{x}_1(t)} u(x,t)\,\mathrm{d}x, && (7.20)\\ \Theta_j(t) &= \int_{\tilde{x}_{j-1}(t)}^{\tilde{x}_{j+1}(t)} u(x,t)\,\mathrm{d}x, \quad j=1,\ldots,N-1, && (7.21)\\ \Theta_N(t) &= \int_{\tilde{x}_{N-1}(t)}^{\tilde{x}_N(t)} u(x,t)\,\mathrm{d}x. && (7.22)\end{aligned}$$

Note that (7.21) and (7.22) do not have a lower limit of zero. The values $\Theta_j(0)$ are known from the initial data. To calculate $\Theta_j(t)$, we begin by constructing $\dot{\Theta}_j(t)$. For ease of explanation we give the explicit formulae for $j=1,\ldots,N-1$ only, but we note that similar formulae hold for $j=0,N$. We differentiate (7.21) using the Leibnitz integral rule to give

$$\dot{\Theta}_j(t) = \frac{\mathrm{d}}{\mathrm{d}t}\int_{\tilde{x}_{j-1}(t)}^{\tilde{x}_{j+1}(t)} u(x,t)\,\mathrm{d}x = \int_{\tilde{x}_{j-1}(t)}^{\tilde{x}_{j+1}(t)} \frac{\partial u}{\partial t}\,\mathrm{d}x + \tilde{u}_{j+1}(t)\tilde{v}_{j+1}(t) - \tilde{u}_{j-1}(t)\tilde{v}_{j-1}(t).$$

Substituting $\frac{\partial u}{\partial t}$ from (7.1) gives

$$\begin{aligned}\dot{\Theta}_j(t) = \int_{\tilde{x}_{j-1}(t)}^{\tilde{x}_{j+1}(t)} S(u,C)\,\mathrm{d}x \quad &+ \quad \tilde{u}_{j+1}(t)\big[\tilde{v}_{j+1}(t) - \tilde{w}_{j+1}(t)\big]\\ &- \quad \tilde{u}_{j-1}(t)\big[\tilde{v}_{j-1}(t) - \tilde{w}_{j-1}(t)\big]. \qquad (7.23)\end{aligned}$$

We use a mid-point approximation of the integral to obtain a discrete form of (7.23) at time $t=t^m$,

$$\dot{\Theta}_j^m = (x_{j+1}^m - x_{j-1}^m)S_j^m + u_{j+1}^m(v_{j+1}^m - w_{j+1}^m) - u_{j-1}^m(v_{j-1}^m - w_{j-1}^m), \qquad (7.24)$$

where $\dot{\Theta}_j^m \approx \dot{\Theta}_j(t^m)$, $j=1,\ldots,N-1$. This equation allows us to determine $\Theta_j^{m+1} \approx \Theta_j(t^{m+1})$ in the same manner that x_j^{m+1} is calculated in Step 4, by the explicit Euler scheme $\Theta_j^{m+1} = \Theta_j^m + \Delta t\dot{\Theta}_j^m$.

Once an approximation to the updated partial masses Θ_j^{m+1} has been determined, the final step for Method A is to recover the solution u_j^{m+1} using a mid-point approximation of (7.21) at time-level $m+1$, i.e.

$$u_j^{m+1} = \frac{\Theta_j^{m+1}}{x_{j+1}^{m+1} - x_{j-1}^{m+1}}, \qquad j=1,\ldots,N-1.$$

As noted above, similar formulae hold for $j=0,N$.

Remark 7.6.1 *The velocity of Method A corresponds to the transformation-based method* *of § 7.5 in the sense that the transformation is effected exactly by the boundary velocity. However, when u is calculated in § 7.5 using a velocity derived from the transformation, a quasi-Lagrangian form of the mass balance equation is used in which the velocity is incorporated using a chain rule. The result is an extra term which cannot be written in divergence form. By contrast, in Method A we have preferred to use an integral approach which already incorporates local conservation.*

7.6.2 Method B

Under this strategy, in Step 3 the velocity of each node is determined by the cell velocity at that node, i.e.

$$v_j^m \quad = \quad w_j^m, \qquad j = 0, 1, \ldots, N.$$

This way of moving the nodes relates to the tumour model more than Method A as it uses local cell information rather than just information from the tumour boundary. Once the mesh velocity has been determined, the new mesh is computed as in Step 4.

In Step 5, as with Method A, we define the partial mass fractions $\Theta_j(t)$ as in (7.21), and follow Method A to completion, noting that $v = w$ at the nodes. In particular, (7.24) reduces to

$$\dot{\Theta}_j^m \quad = \quad (x_{j+1}^m - x_{j-1}^m) S_j^m, \qquad j = 1, \ldots, N-1,$$

since the terms in the square brackets of (7.23) are zero for $v_j^m = w_j^m$. Note that this method corresponds with the mass balance equation (7.4) over arbitrary subintervals,

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\tilde{x}_{j-1}(t)}^{\tilde{x}_{j+1}(t)} u(x,t) \, \mathrm{d}x = \int_{\tilde{x}_{j-1}(t)}^{\tilde{x}_{j+1}(t)} S(u,C) \, \mathrm{d}x.$$

Therefore this is the same as the moving mesh method described in §3.3.

7.6.3 Method C

Method C moves the nodes so as to conserve local mass fractions. Like Method B, this method also uses a feature of the model to move the nodes in such a way that information about the distribution of cells within the tumour is carried in time. We refer to the method in §3.2 for

$$\mathcal{G} \equiv S(u,C) - \frac{\partial}{\partial x}(wu), \tag{7.25}$$

and $a(t) = 0$, $b(t) = b(t)$. Let the total mass be

$$\theta(t) = \int_0^{b(t)} u(x,t)\,\mathrm{d}x,$$

as given by (3.19). We define c_j to be the mass fraction given by (3.19), so that

$$c_j = \frac{1}{\theta(t)} \int_0^{\tilde{x}_j(t)} u(x,t)\,\mathrm{d}x, \tag{7.26}$$

and calculate $\tilde{x}_j(t)$ such that c_j remains constant with respect to time.

The total mass θ will be required in order to approximate u, so we first determine $\dot{\theta}$ by substituting (7.25) into (3.22) giving,

$$\dot{\theta}(t) \quad = \quad \int_0^{b(t)} S(u,C) - \frac{\partial}{\partial x}(wu)\,\mathrm{d}x + u_b v_b - u_0 v_0.$$

Using the boundary conditions (7.6)–(7.7), gives

$$\dot{\theta}(t) \quad = \quad \int_0^{b(t)} S(u,C)\,\mathrm{d}x. \tag{7.27}$$

It is worth noting that equation (7.27) corresponds exactly to the global mass balance result (7.4).

Equation (7.27) can be approximated directly once Step 1 has been carried out. We define the approximation $\dot{\theta}^m \approx \dot{\theta}(t^m)$ and apply a trapezoidal rule approximation to (7.27),

$$\dot{\theta}^m = \sum_{j=0}^{N-1} \frac{1}{2}(x_{j+1}^m - x_j^m)(S_{j+1}^m + S_j^m). \tag{7.28}$$

The updated total mass $\theta^{m+1} \approx \theta(t^{m+1})$ is then found using (7.28) and the same time-stepping approach used in Step 4, i.e. $\theta^{m+1} = \theta^m + \Delta t \dot{\theta}^m$.

To derive an expression for the mesh velocity, we substitute (7.25) into (3.23) giving,

$$\tilde{v}_j(t) = \frac{1}{\tilde{u}_j(t)} \left(\dot{\theta}(t) c_j - \int_0^{\tilde{x}_j(t)} S(u,C) - \frac{\partial}{\partial x}(wu)\,\mathrm{d}x \right).$$

Using the boundary condition $u_0^m = v_0^m = 0$ (from (7.6)), gives

$$\tilde{v}_j(t) = \frac{1}{\tilde{u}_j(t)} \left(\dot{\theta}(t) c_j - \int_0^{\tilde{x}_j(t)} S(u,C)\,\mathrm{d}x \right) - \tilde{w}_j(t). \tag{7.29}$$

for $\tilde{u}_j(t) = u(\tilde{x}_j, t) \neq 0$. We use the composite trapezoidal rule on the integral to obtain a discrete form of (7.29) at time $t = t^m$,

$$v_j^m = \frac{c_j \dot{\theta}^m}{u_j^m} - \frac{1}{u_j^m} \sum_{i=0}^{j-1} \frac{1}{2}(x_{i+1}^m - x_i^m)(S_{i+1}^m + S_i^m) + w_j^m. \tag{7.30}$$

Using (7.30), the new mesh x_j^{m+1} is computed as in Step 4. To approximate the updated solution u_j^{m+1} in Step 5, we use (3.25),

$$u_j^{m+1} = \frac{\theta^{m+1}}{\theta^0} \frac{\left(x_{j+1}^0 - x_{j-1}^0\right)}{\left(x_{j+1}^{m+1} - x_{j-1}^{m+1}\right)} u_j^0.$$

7.7 Numerical Results

In this section we solve the tumour growth model numerically using the methods of §7.5 and §7.6, and compare the outcomes from each approach. In our experiments we used two sets of parameters from [22], which were chosen so as to focus on the qualitative nature of the model equations. A purpose of [22] was to examine the effect of altering the tension constant (by altering k and μ). To compare our moving mesh methods to the commonly used fixed mesh method, we choose the parameters from [22] that correspond to plots of u, v and b over time. Both sets of parameters take

$$Q = 0.5, \quad Q_1 = 0, \quad s_1 = s_4 = 10, \quad s_2 = s_3 = 0.5, \quad u^0(x) = u^* = 0.8, \tag{7.31}$$

with

$$k = 1, \quad \mu = 1, \quad u_{\min} = 0.8, \tag{7.32}$$

in the first case, and

$$k = 0.25, \quad \mu = 0.25, \quad u_{\min} = 0.6, \tag{7.33}$$

in the second case. The first case does not include the effects of cellular attraction, whilst the second case does. Furthermore, the second case has smaller k and μ than the first case, which corresponds to a larger tension constant. Figures 7.3(a)–7.4(d) show results obtained with the method described in §7.5, with $N = 80$, $\Delta t = 7.5 \times 10^{-3}$ and final time $t = 75$, i.e. 10,000 time steps. Figure 7.3 uses parameters (7.31)–(7.32) and display a travelling wave solution. Figure 7.4 uses the second set of parameters, (7.31) and (7.33), and show the tumour radius settling to a steady state. Figures 7.3–7.4 closely resemble the results shown in [22] (however, results for the nutrient concentration were not included in [22]).

Next we examine the convergence of the moving mesh methods of §7.6 for the parameter set (7.31) and (7.32), as N increases and Δt decreases. We solve for $t \in [0, 4]$ and compute results for $N = 10 \times 2^{\hat{N}-1}$, $\hat{N} = 1, \dots, 6$. In order to compare results for different values of $\hat{N}$, we denote the points of the mesh for a particular value of $\hat{N}$ by $x_{j,\hat{N}}(t)$, $j = 0, \dots, (10 \times 2^{\hat{N}-1})$. We then compute both $\tilde{x}_j(4)$ and $\tilde{u}_j(4)$ at $j = 2^{n-1}i$ for each $i = 0, \dots, 10$ as $\hat{N}$ increases. The values of $\tilde{x}_{2^{n-1}i}(4)$ and $\tilde{u}_{2^{n-1}i}(4)$ at these points are represented by $x_{2^{\hat{N}-1}i,\hat{N}}$ and $u_{2^{\hat{N}-1}i,\hat{N}}$ respectively. To balance the spatial and temporal errors, and recalling that we have used explicit Euler time-stepping, we choose $\Delta t = \mathcal{O}\left(\frac{1}{N^2}\right)$, precisely $\Delta t = \frac{1}{50(4^{\hat{N}})}$. We take the results computed with $\hat{N} = 6$ (i.e. $N = 320$) as our reference mesh and solution. We anticipate that the pointwise 'errors' $|u_{32i,6} - u_{2^{\hat{N}-1}i,\hat{N}}|$ and $|x_{32i,6} - x_{2^{\hat{N}-1}i,\hat{N}}|$ will decrease as $\hat{N}$ increases, for each $i = 0, \dots, 10$.

As a measure of the errors, we calculate

$$E_N(u) = \sqrt{\frac{\sum_{i=0}^{10}(u_{32i,6} - u_{2^{\hat{N}-1}i,\hat{N}})^2}{\sum_{i=0}^{10}(u_{32i,6})^2}}, \qquad E_N(\tilde{x}) = \sqrt{\frac{\sum_{i=0}^{10}(x_{32i,6} - x_{2^{\hat{N}-1}i,\hat{N}})^2}{\sum_{i=0}^{10}(x_{32i,6})^2}},$$

for $\hat{N} = 1, \dots, 4$ (i.e. $N = 10, 20, 40, 80$). We investigate the hypothesis that

$$E_N(u) \sim \frac{1}{N^p} \qquad \text{and} \qquad E_N(\tilde{x}) \sim \frac{1}{N^q},$$

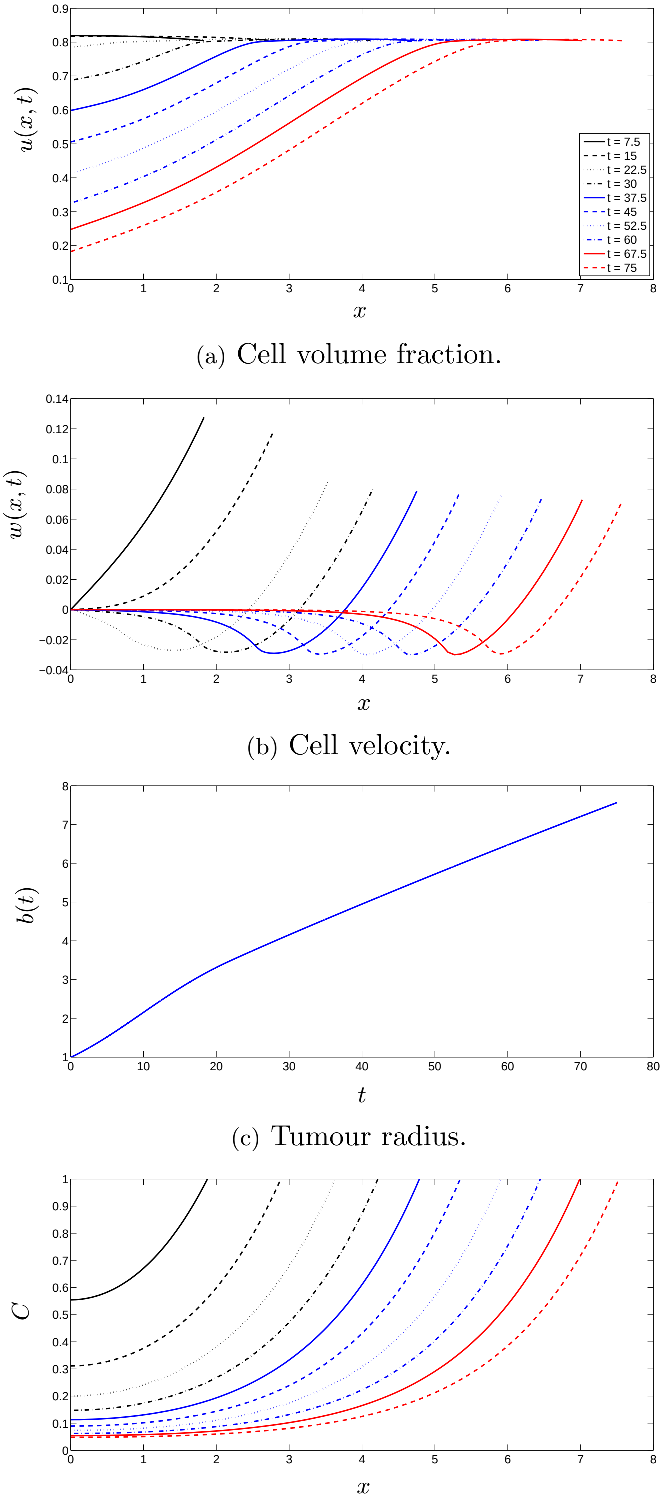
for large N, where p and q are the estimated orders of convergence for $E_N(u)$ and $E_N(\tilde{x})$ approximated respectively by

$$p_{2N} = -\log_2\left(\frac{E_{2N}(u)}{E_N(u)}\right) \qquad q_{2N} = -\log_2\left(\frac{E_{2N}(\tilde{x})}{E_N(\tilde{x})}\right).$$

Since each step of our scheme is second order in space and first order in time, and recalling that $\Delta t = \mathcal{O}\left(\frac{1}{N^2}\right)$, we might expect to see $p, q \approx 2$. Convergence results are shown in Table 7.1. We see that $E_N(u)$ and $E_N(\tilde{x})$ decrease as N increases for each of the moving mesh methods. This strongly suggests that as the number of nodes increases, both the solution u and the position of the nodes $\tilde{x}_j$ are converging. For Methods A and B, the p-values presented in this table indicate superlinear convergence of u, and the q-values suggest second-order convergence of $\tilde{x}$. For Method C, the p and q values suggest second-order convergence of both u and $\tilde{x}$.

Having established convergence of our moving mesh schemes we now compare the numerical results from the methods of §7.6 with those of the method described in §7.5.

We generate results using the parameters detailed in (7.31) and (7.32). All three



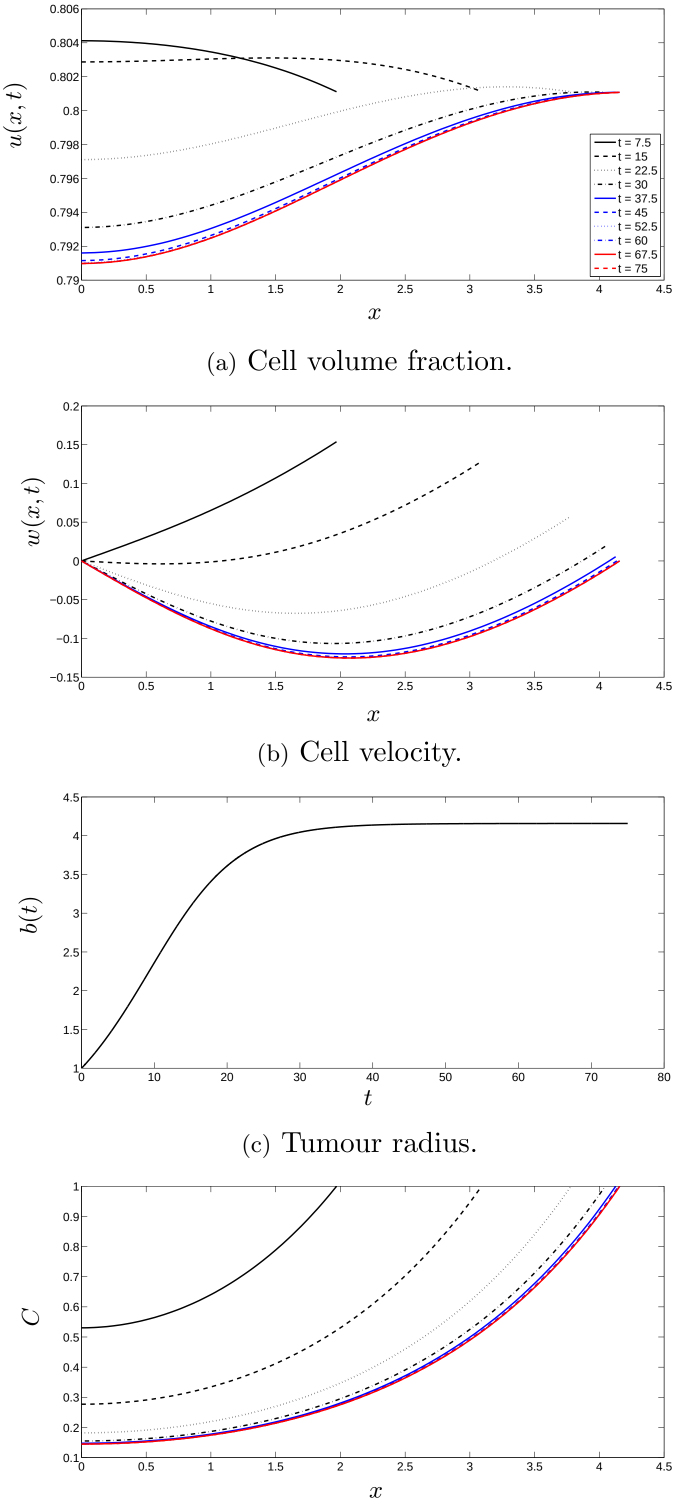
(a) Cell volume fraction.

(b) Cell velocity.

(c) Tumour radius.

(d) Nutrient concentration.

Fig. 7.3: The fixed numerical mesh method and parameter set (7.31)–(7.32), legend in Figure 7.3(a).



(a) Cell volume fraction.

(b) Cell velocity.

(c) Tumour radius.

(d) Nutrient concentration.

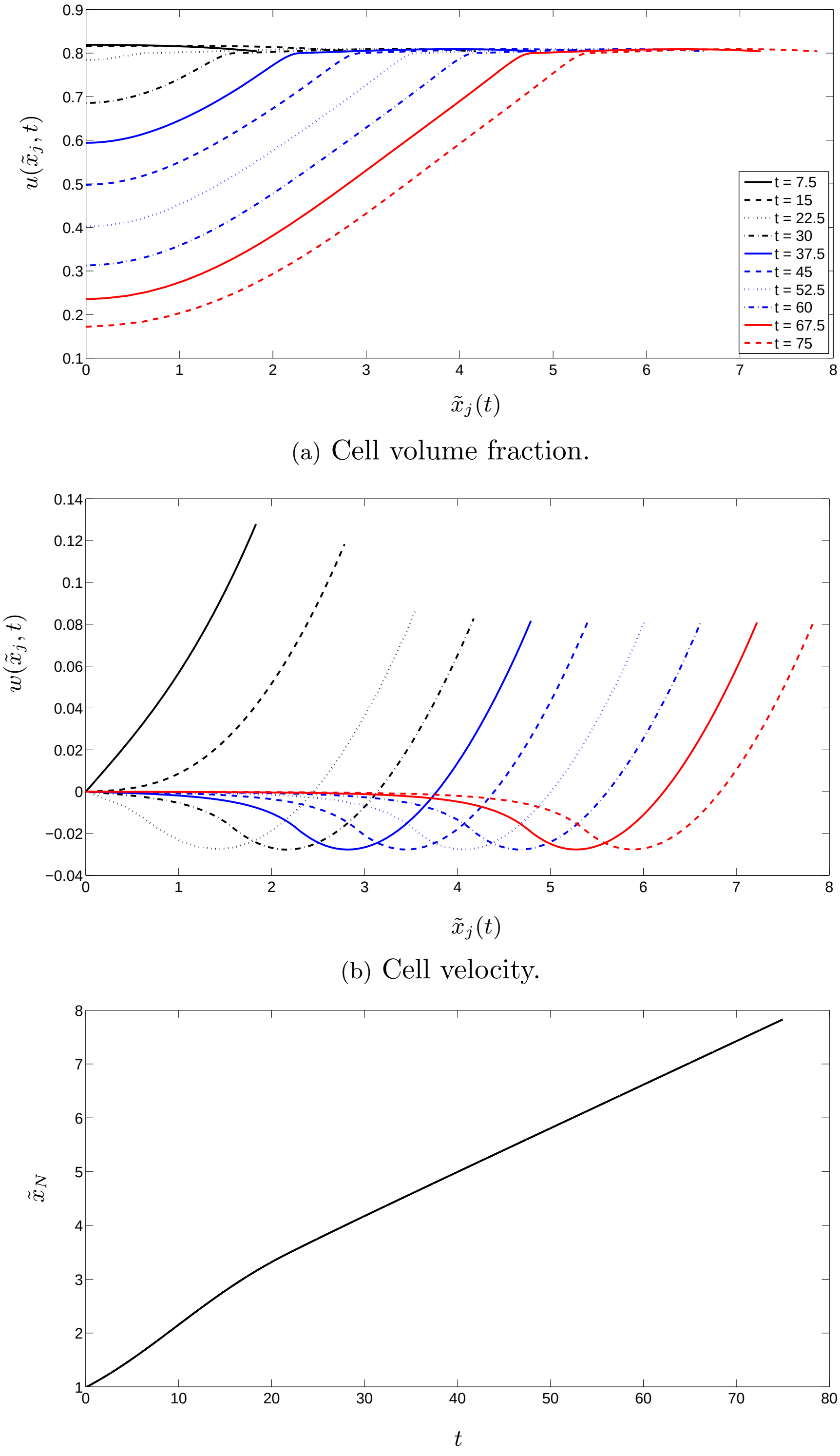
Fig. 7.4: The fixed numerical mesh method and parameter set (7.31) and (7.33), legend in Figure 7.4(a).

Method	N	$E_N(u)$	p_N	$E_N(\tilde{x})$	q_N
A	10	2.034×10^{-4}	-	1.275×10^{-5}	-
	20	8.346×10^{-5}	1.3	3.306×10^{-6}	1.9
	40	3.547×10^{-5}	1.2	8.478×10^{-7}	2.0
	80	1.471×10^{-5}	1.3	2.050×10^{-7}	2.0
B	10	2.299×10^{-4}	-	6.207×10^{-4}	-
	20	9.293×10^{-5}	1.3	1.109×10^{-4}	2.5
	40	3.891×10^{-5}	1.3	3.043×10^{-5}	1.9
	80	1.600×10^{-5}	1.3	7.224×10^{-6}	2.1
C	10	1.448×10^{-5}	-	1.819×10^{-5}	-
	20	3.645×10^{-6}	2.0	1.944×10^{-6}	3.2
	40	8.807×10^{-7}	2.0	7.148×10^{-7}	1.5
	80	2.090×10^{-7}	2.1	1.880×10^{-7}	1.9

Table 7.1: Relative errors for u and $\tilde{x}$ with rates of convergence using the explicit Euler time-stepping scheme.

methods were investigated with $N = 80$, $\Delta t = 7.5 \times 10^{-3}$, and final time $t = 75$, i.e. 10,000 time-steps. Each of Methods A and C produce very similar results, so only the results from Method C and Method B are plotted below. Figure 7.5 is due to Method C and displays the same travelling wave characteristics as the results in [22] for the same parameters (closely resembling Figures 7.3(a)–7.3(c)). The value of u near the free boundary remains fairly constant, and u at the centre of the tumour decreases at a steady rate as time increases. The velocity peaks near the boundary, but the velocity at the boundary appears to stay constant with respect to time for $t \geq 37.5$. This coincides with the tumour radius growing steadily, Figure 7.5(c). The minima are subtly different to that of [22]; the troughs in Figure 7.3(b), which resemble those in [22], are slightly less rounded than those shown in Figure 7.5(b). Interestingly, Method A (a locally conservative version of the method in §7.5) also presented rounder minima, identical to those in Figure 7.5(b).

Figure 7.6 show that Method B appears to behave like Method A and C (and [22]) at early times. However, after approximately $t = 45$, u appears to grow at the boundary, and no longer decreases at a regular rate at the centre of the tumour. Furthermore, the velocity at the boundary decreases considerably, with the tumour radius almost at a steady state at $t = 75$. This behaviour is not apparent in [22], nor from Methods A and C. The plots from Method B are less smooth, despite the same number of nodes being used for each method. There is a considerable kink in u and w for $t = 45$ which appears to dampen at later times. The solution u does not drop below 0.4 at the centre of the tumour, even for $t = 100$ (not shown here). This erratic behaviour remains with a smaller Δt, and when using an adaptive second and third order Runge-Kutta method for the time-stepping (see Remark 7.7.1 below), suggesting that this behaviour is due to the choice of the velocity in



(a) Cell volume fraction.

(b) Cell velocity.

(c) Tumour radius.

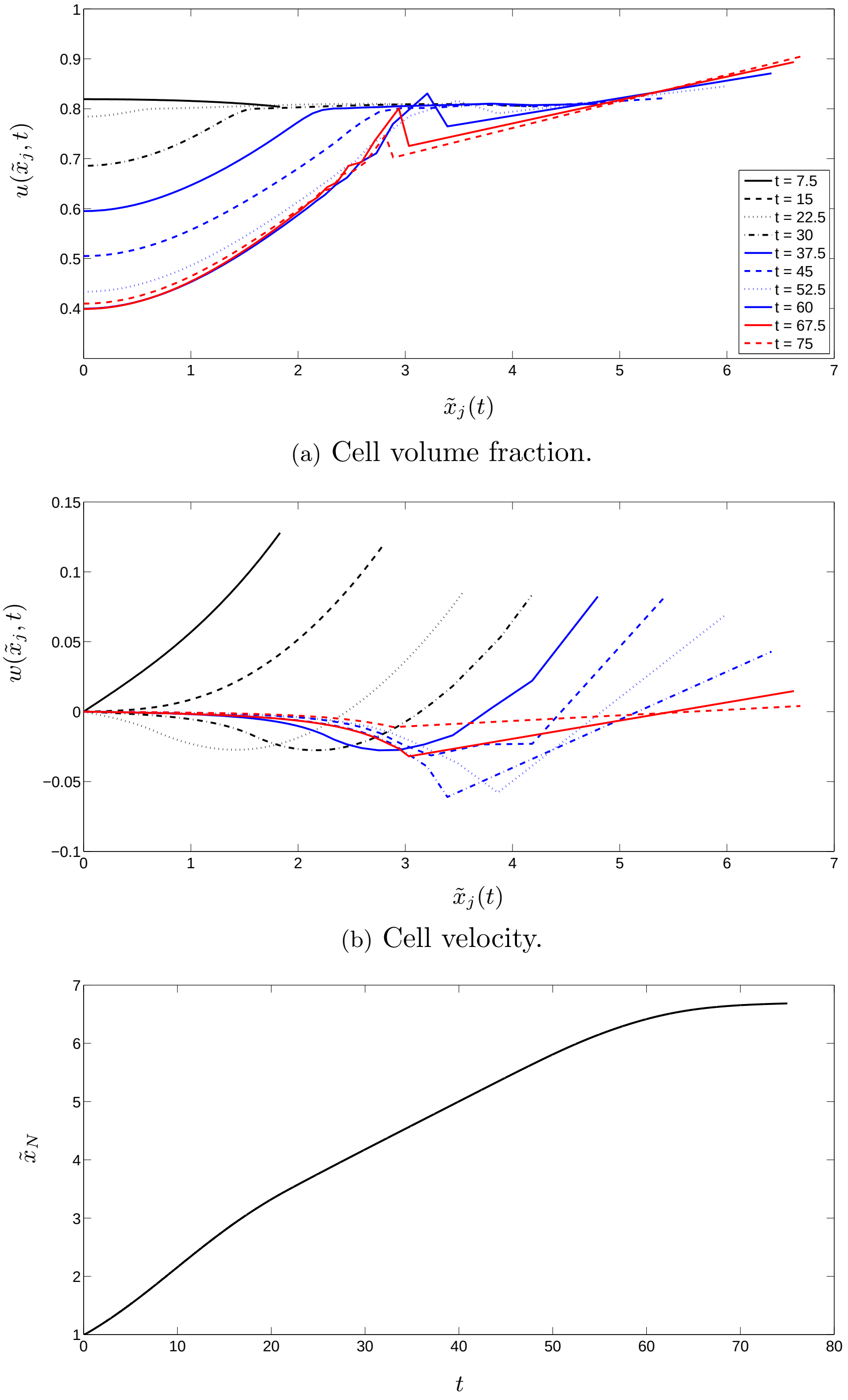
Fig. 7.5: Method C and parameter set (7.31)–(7.32), legend in Figure 7.5(a).

the numerical method. The processes of Method A and Method B are very similar, and because Method A behaves as in Figure 7.5, it is reasonable to conclude that tracking the cell velocity with the mesh nodes, as in Method B, results in the mesh becoming too coarse in some areas, and too fine in others. This is a problem that could be compounded over time, especially in the area where the cell velocities vary between positive and negative; resulting in nodes moving in opposite directions, leaving a considerable gap in between. Indeed if we look at Figure 7.6(b) for $t = 75$, we see that the velocity is mostly negative, so that most of the nodes are moving to the left.

As a further example, we use the parameter set (7.31) and (7.33), and again present results for the method of §7.5 and the moving mesh methods in §7.6. Once the steady state is reached at $t \approx 40$, all cells within the region have negative velocity, i.e. the cells are moving inwards. The comparisons between the methods had similar outcomes: the results for Methods A and C (Figure 7.7) resembled the results in [22] (as shown in Figures 7.4(a)–7.4(c)); Method B moves the nodes evenly for early times, but once negative spatial velocities occur, the nodes become clustered to the left, as shown in Figure 7.8. When the tumour radius settles to a steady state, the internal cells continue moving. This feature means that the mesh for Method B never settles to a steady state, whereas the meshes for Methods A and C do.

Finally, we examine exactly how the mesh moves for each of the different moving mesh methods. We take the parameters that produce a steady travelling-wave profile, (7.31) and (7.32). By definition, the nodes with Method A remain equally spaced over time, and move to the right uniformly with the tumour growth, as shown in Figure 7.9(a). The mesh for Method B, Figure 7.9(b), begins by spreading out fairly equally. However, at later times when negative velocities are introduced, the nodes cluster nearer the centre of the tumour. Indeed, it can be seen that most nodes will initially move out with the tumour growth, but then return to the tumour centre. The node at the boundary is then significantly separated from the others, causing unsatisfactory coarseness at the edge. When the nodes are moved by Method C, Figure 7.9(c), the nodes behave similar to the nodes of Method A for $t < 30$. For larger times, the nodes near the tumour centre spread. We would expect the spread to be more prominent as the tumour grows, i.e. the nodes naturally spread where u is low, and cluster where u is larger. Moreover, each node only moves to the right as the tumour grows. When comparing Figures 7.9(a) and 7.9(c) it becomes apparent why they produce nearly the same results, especially for $t \leq 30$.

Remark 7.7.1 *For the moving mesh methods we also considered using a time-stepping scheme based on an adaptive second and third order predictor-corrector Runge-Kutta method, which chooses the time step automatically to minimise the error (specifically, we used*

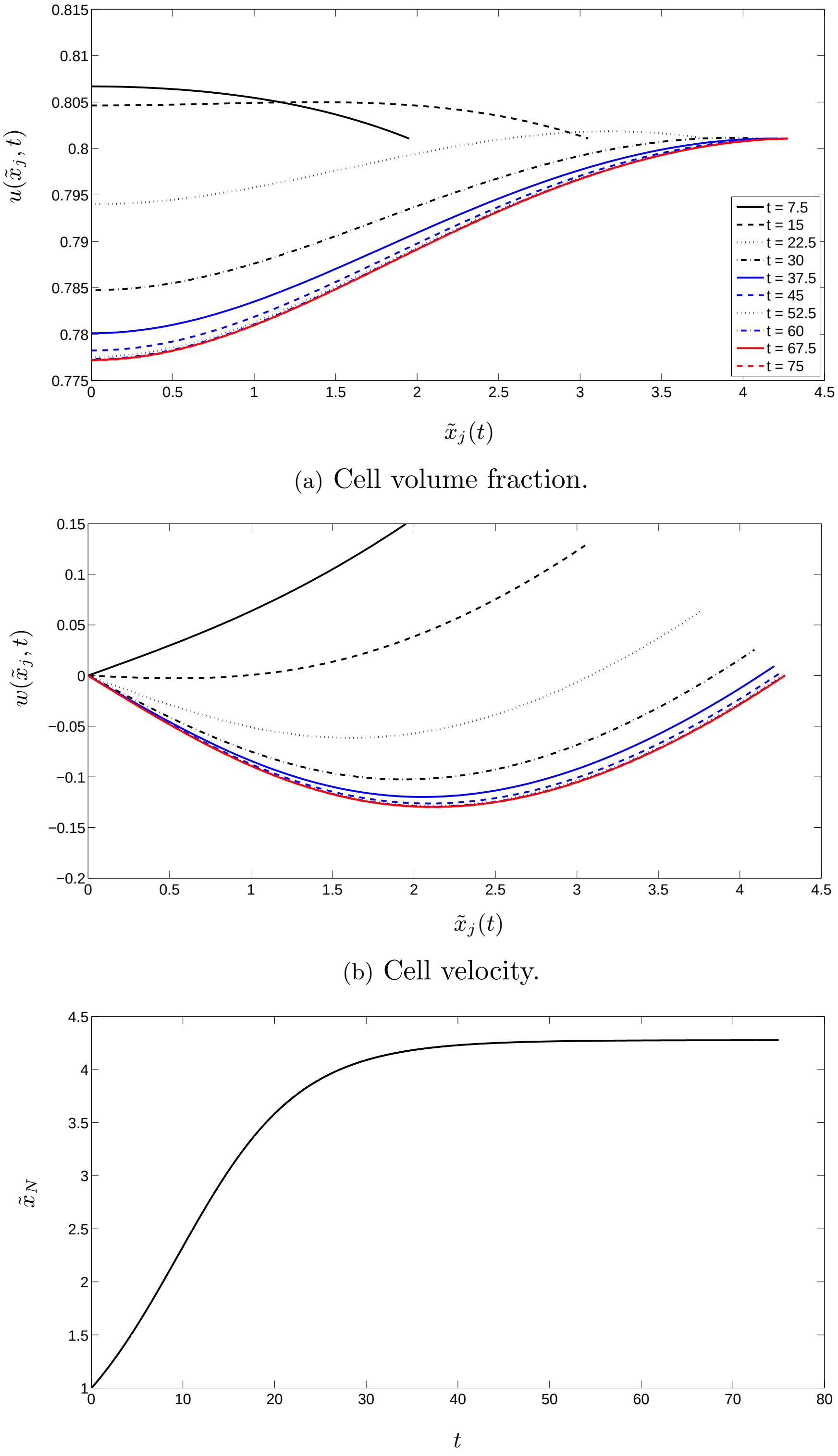


(a) Cell volume fraction.

(b) Cell velocity.

(c) Tumour radius.

Fig. 7.6: Method B and parameter set (7.31)–(7.32), legend in Figure 7.6(a).

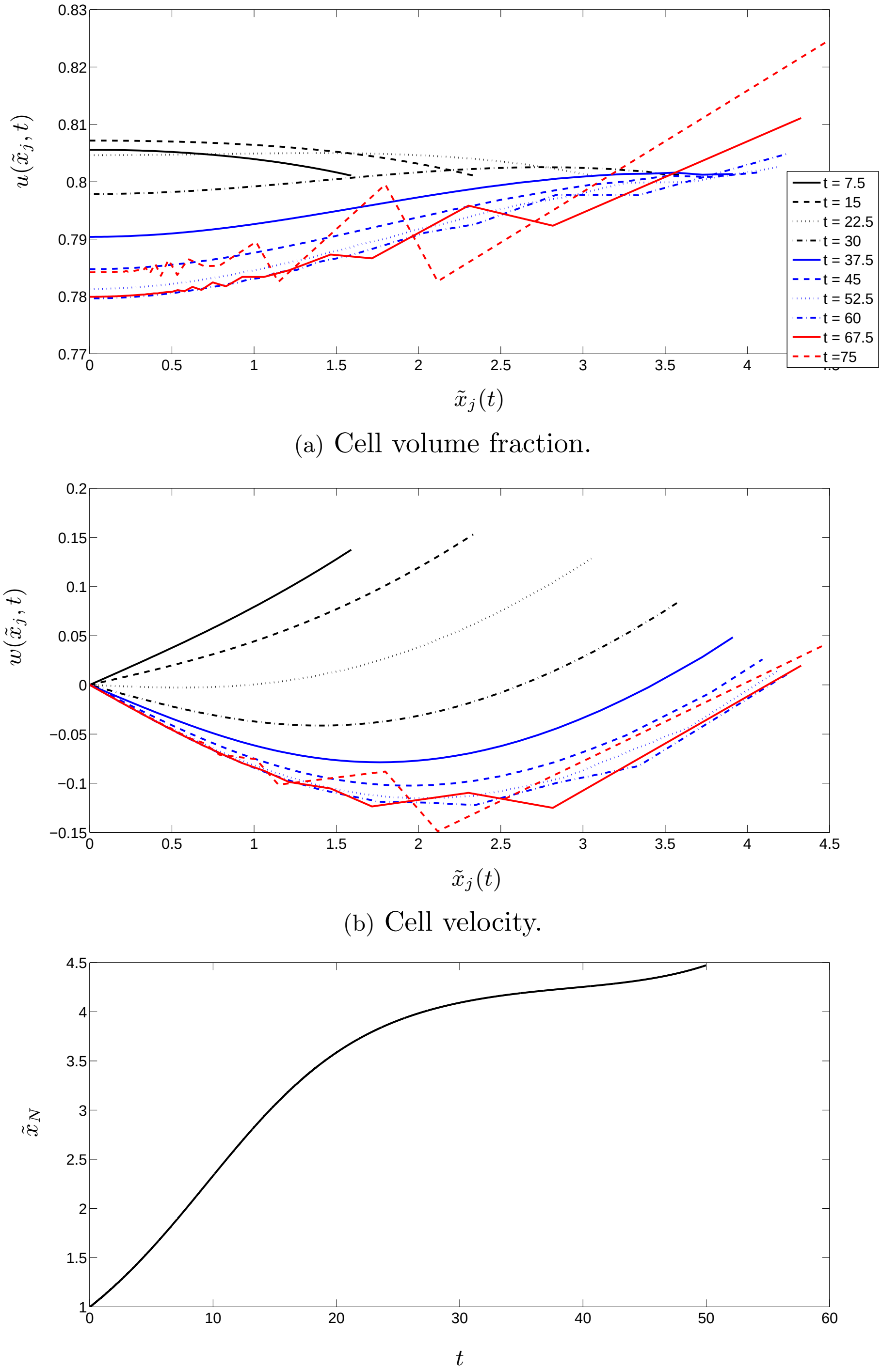


(a) Cell volume fraction.

(b) Cell velocity.

(c) Tumour radius.

Fig. 7.7: Method C and parameter set (7.31) and (7.33), legend in Figure 7.7(a).

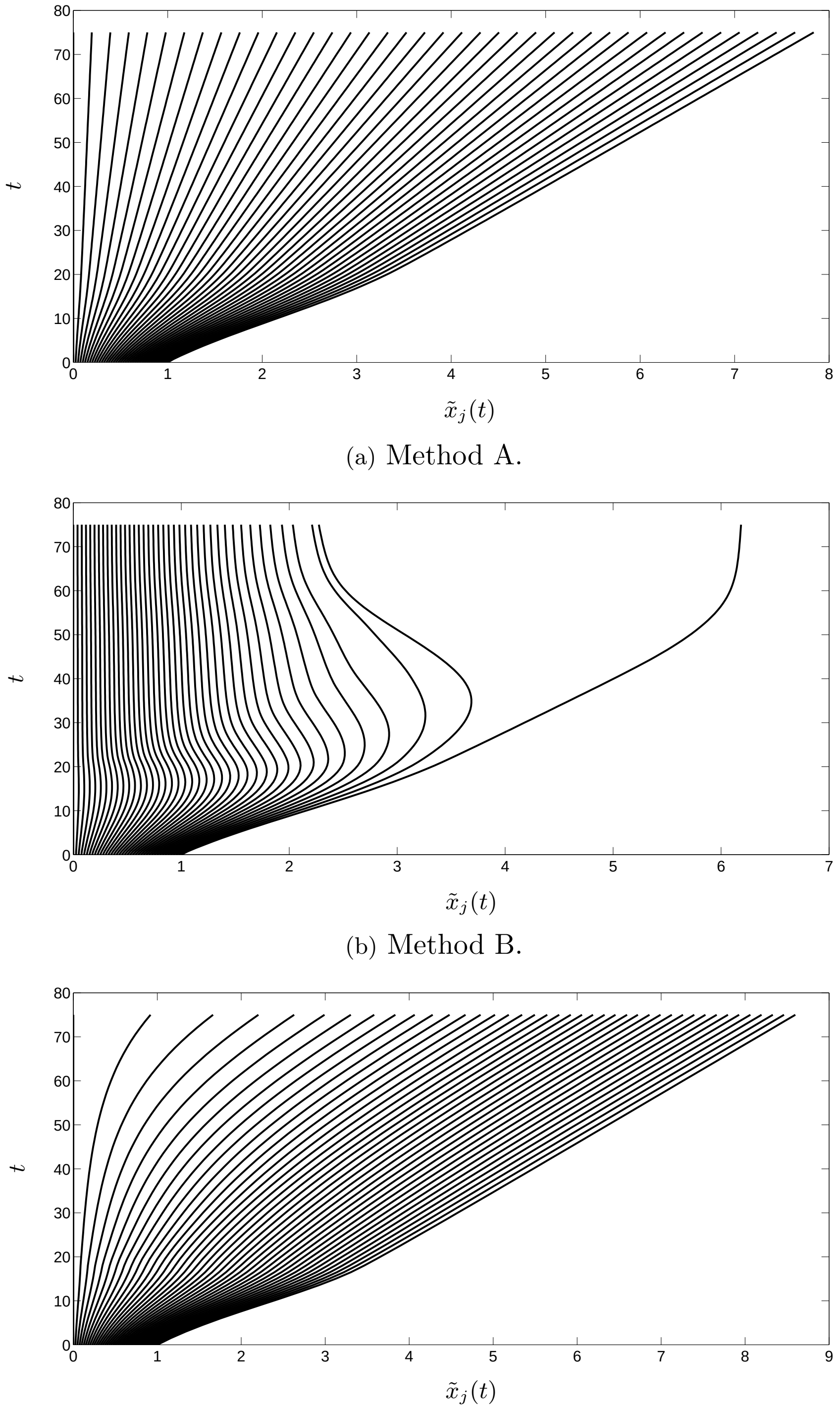


(a) Cell volume fraction.

(b) Cell velocity.

(c) Tumour radius.

Fig. 7.8: Method B and parameter set (7.31) and (7.33), legend in Figure 7.8(a).



(a) Method A.

(b) Method B.

(c) Method C.

Fig. 7.9: The position of nodes, parameter set (7.31) and (7.33).

ODE23 in Matlab). When using this scheme, we took a maximum $\Delta t = \mathcal{O}\left(\frac{1}{N}\right)$ *to balance the spatial and temporal errors, precisely* $\max \Delta t = \frac{1}{50(2^n)}$. *The results from the Runge-Kutta method were very similar to results from the explicit Euler time-stepping scheme, indicating that our approach is robust to different time-stepping approaches, and is not particularly stiff.*

7.8 Summary and conclusions for the tumour growth problem

We have numerically solved the non-dimensionalised form of an avascular tumour growth model given in [22] using three different moving mesh methods. Working with the original non-dimensionalised form of the model, we have replicated the results of [22] and presented three different velocity-based approaches to move the mesh. The different approaches to define the mesh velocity are: (A) proportional to the boundary movement; (B) following the cell velocity; (C) conserving local mass fractions. To advance in time, each of the three methods used either explicit Euler time-stepping or adaptive second and third order Runge-Kutta formulas. Each method, with explicit Euler time-stepping, appears to be convergent for small times. Methods A and C continue to work well for larger times and replicate results in [22], but Method C has the added advantage that the nodes move in a manner that preserves a feature of the model, specifically local mass fractions resulting in higher resolution at the boundary. However, care is required when choosing a feature of the model to determine the mesh velocity, as evidenced by the poor resolution apparent when using Method B over longer times. Method C is an especially effective method when solving problems with self-similar solutions as it preserves similarity.

In the two-phase model studied here the outer boundary is accurately followed. More recent three-phase models that take into account proliferating, quiescent and necrotic cells can be treated in a similar way, even though these models cannot be reduced to the study of a single component such as u. However, in a two-phase situation the necrotic core can possibly be modelled as a separate inner region between inner and outer moving boundaries.

A paper detailing the results in this chapter has been submitted to ‘Mathematical and Computer Modelling’ [65].

The next chapter summarises our work on moving meshes.

8

Summary and Further Work

Work on moving meshes has evolved considerably over recent years, becoming a versatile tool to accurately simulate a wide range of problems. The key advantage of a moving mesh is its ability to adjust its distribution to focus on areas of interest, such as a moving boundary or blow-up. In this thesis we have discussed one such method, a finite difference moving mesh method which is well-adapted to solving one-dimensional nonlinear IBVPs. In most cases the velocity was determined by considering the partial integrals of the solution

$$\int_{a(t)}^{\tilde{x}_j(t)} u \, \mathrm{d}x, \tag{8.1}$$

and either keeping them constant or balancing them with one of the features of the PDE. In §3.1 we considered problems that conserve mass, and set the partial masses (8.1) to be constant. In §3.2 we generalised upon this approach for problems that do not conserve mass, and set the relative partial masses

$$\frac{\int_{a(t)}^{\tilde{x}_j(t)} u \, \mathrm{d}x}{\int_{a(t)}^{b(t)} u \, \mathrm{d}x},$$

to be constant. This strategy is related to the GCL method and is similar to that used by Baines, Hubbard and Jimack for their moving mesh finite element algorithm [5]. We also considered balancing the partial masses (8.1) with a source term, where appropriate.

We applied these methods to a number of moving boundary problems to investigate the effectiveness of this moving mesh approach. The problems we numerically solved to demonstrate how our moving mesh approach increased in complexity, initially looking at problems which conserve mass: the PME and Richards' equation, both of which are fluid flow problems. Then we looked at a problem with a variable mass: the Crank-Gupta problem, which is used to model oxygen-diffusion through tissue. Lastly, we considered an avascular tumour growth model, which is a system for which the mass increases over time. This has three PDEs (two quasi-steady) which need to be updated at each time-level. The quasi-steady PDEs were solved using finite difference on an irregular mesh, but this process did not compromise the moving mesh method. We summarise the application of each moving mesh approach in turn, and then discuss the time-stepping schemes used.

Preserving mass fractions

Preserving mass fractions was applied in all the problems. We examined the accuracy in all cases and found that the numerical solution converged with roughly second-order accuracy. Furthermore, for the Crank-Gupta problem and the tumour growth model, we found that preserving mass fractions can lead to a higher resolution at the boundary, which is desirable.

However, the radial case for the Crank-Gupta problem highlighted problems that may occur when estimating the boundary position. Generally, if the boundary positions are not given, we calculated the boundary position with a polynomial extrapolation from the points near the moving boundary. The Crank-Gupta problem approaches the outer boundary smoothly with a zero derivative, making extrapolation inaccurate. This is overcome in the one-dimensional case by considering the asymptotic behaviour of the solution near the boundary, but this was not suitable for the radial Crank-Gupta problem. This is of little concern as regards the oxygen-diffusion model, since our model was one-dimensional (even though in reality the problem is three-dimensional). However, it highlights that an extrapolation may not always be suitable, and care must be taken when estimating the boundary position.

Using an alternative mass balance

As an alternative, the method given in §3.3 was applied to the Crank-Gupta problem and the tumour growth model. For Richards' equation in §5.6 we considered a slight variation of the method given in §3.3. The method in §3.3 set the partial masses equal to the source term, and the method in §5.6 set the partial masses equal to the flux term. When balancing the partial masses with a term from the PDE the results are less satisfactory than setting them to a constant. We saw that balancing the partial masses to the flux term of Richards' equation provided results that initially appear satisfactory, but when compared to the mass conserving approach we notice that the alternative method is less accurate. For the Crank-Gupta problem, balancing the partial masses with the source term led to severe inaccuracies in the boundary behaviour. For the tumour growth problem, balancing the source term with the partial masses is equivalent to using the cell velocity to define the node movement. This led to inaccuracies when cells displayed negative velocities (indicating that the cells were moving inwards). Our work suggests that careful consideration is required when choosing a feature to balance with the partial masses, and any inaccuracies which result from a bad choice will be unique to the problem being numerically solved.

Time-stepping schemes

Throughout this thesis we have used an explicit Euler time-stepping scheme. Other explicit time-stepping schemes we have used are the Runge-Kutta predictor-corrector methods built into Matlab (ODE23, ODE45, ODE15s). There was little difference in the results from all the Matlab solvers, indicating that none of the problems lead to a stiff system of differential equations for the $\tilde{x}_j(t)$. We found that all explicit time-stepping schemes produced accurate and stable results, with no mesh tangling, provided that sufficiently small time-steps were taken.

The semi-implicit scheme we proposed in §3.4 ensures that the mesh does not tangle. We found this approach to be suitable for the PME, allowing the time-step to be larger than when an explicit time-stepping scheme is used. However, care is still required when taking a larger time step to ensure accuracy. The semi-implicit time-stepping scheme is less versatile than the explicit Euler time-stepping scheme since not all mesh velocity equations (such as Richards' equation) can be written in the required form

$$v_j^m = \frac{1}{\Delta x_j^m}\Big(\phi_{j+}^m - \phi_{j-}^m\Big).$$

However, Richards' equation showed that we can still use a semi-implicit scheme that is not in the form above, but Theorem 3.4.1 is not satisfied. For this equation, we adapted the moving mesh approach so as to balance the partial masses with the flux term. This allowed us to use a semi-implicit time-stepping scheme which satisfied Theorem 3.4.1, but we found it to be less accurate since the partial masses are updated explicitly, whilst the mesh velocities are updated semi-implicitly, suggesting that updating them simultaneously is necessary. This concept is more apparent with the Crank-Gupta problem where we noticed inaccuracies at the inner boundary.

Conclusions and further work

We conclude that the mass-conserving approach, with an explicit time-stepping scheme, is accurate for a range of problems. We note that care is required when calculating the boundary velocities, and when implementing a semi-implicit time-stepping scheme. It may be desirable to apply a semi-implicit time-stepping scheme when numerically solving the tumour growth model since the solution over larger time may be required.

We considered some PME properties in details, but we did not investigate ordering of solutions and the existence of attractors. These properties are known analytically, and we would have liked to replicate them numerically. This is an area of current investiagtion for which a paper is in preparation.

The tumour growth model demonstrates that the method can be used on more complex problems arising from various mathematical models. There exists many problems, such as a three phase model of tumour growth [96], the slime mould model [31] and cell motility [40], which are systems where two or more variables are coupled. A natural progression from this thesis would be to apply the moving mesh method to these problems by choosing a preferred variable to determine the mesh movement.

Bibliography

[1] Ahamadi, M. and Harlen, O.G. (2008) A Lagrangian finite element method for simulation of a suspension under planar extensional flow. *J. Comput. Phys.* **227** 7543-7560.

[2] Araujo, R.P. and McElwain, D.L.S. (2004) A history of the study of solid tumour growth: The contribution of mathematical modelling. *Bull. Math. Biol.* **66** 1039–1091.

[3] Baer, T.A., Cairncross, R.A., Schunk, P.R., Rao, R.R. and Sackinger, P.A. (2000) A finite element method for free surface flows of incompressible fluids in three dimensions. Part II. Dynamic wetting lines. *Int. J. Numer. Meth.Fl.* **33** 405-427.

[4] Baines, M.J. (1994) Moving Finite Elements. *Oxford University Press.*

[5] Baines, M.J., Hubbard, M.E. and Jimack, P.K. (2005) A moving mesh finite element algorithm for the adaptive solution of time-dependent partial differential equations with moving boundaries. *Appl. Numer. Math.* **54** 450–469.

[6] Baines, M.J., Hubbard, M.E. and Jimack, P.K. (2005) A moving mesh finite element algorithm for fluid flow problems with moving boundaries. *Int. J. Numer. Meth. Fl.* **47** 1077–1083.

[7] Baines, M.J., Hubbard, M.E., Jimack, P.K. and Jones, A.C. (2006) Scale-invariant moving finite elements for nonlinear partial differential equations in two dimensions. *Appl. Numer. Math.* **56** 230-252.

[8] Baines, M.J., Hubbard, M.E., Jimack, P.K. and Mahmood, R. (2009) A moving-mesh finite element method and its application to the numerical solution of phase-change problems. *Commun. Comput. Phys.* **6** 595–624.

[9] Baines, M.J., Hubbard, M.E. and Jimack, P.K. (2011) Velocity-based moving mesh methods for nonlinear partial differential equations. *Commun. Comput. Phys.* To appear.

[10] Barari, A., Omidvar, M., Ghotbi, A.R. and Ganji, D.D. (2009) Numerical analysis of Richards' problem for water penetration in unsaturated soils. *Hydrol. Earth Syst. Sci.* **6** 6359-6385.

[11] Barenblatt, G.I. (1996) Scaling, self-similarity, and intermediate asymptotics. *Cambridge University Press.*

[12] Barenblatt, G.I. (1952) On some unsteady motions of fluids and gases in a porous medium. *Prikladnaya Matematika i Mekhanika. (Translated in J. Appl. Math. Mech.)* **6** 67–78.

[13] Barlow, A. (2008) A compatible finite element multi-material ALE hydrodynamics algorithm. *Int. J. Numer. Meth. Fluids* **56** 953–964.

[14] Becket, G., Mackenzie, J.A. and Roberston, M. L. (2001) A moving mesh finite element method for the solution of two-dimensional Stefan problems. *J. Comput. Phys.* **168** 500-518.

[15] Berg, M.de, Cheong, O., Kreveld, M.van and Overmars, M. (2008) Computational Geometry: Algorithms and Applications. *Springer-Verlag.*

[16] Blake, K.W. (2001) Moving mesh methods for non-linear parabolic partial differential equations. *Ph.D Thesis, University of Reading, UK.*

[17] Blake, K. and Baines, M.J. (2002) A moving mesh method for non-linear parabolic problems. *Numerical Analysis Reports, 2/2002, University of Reading, UK.*

[18] Blowey, J.F., King, J.R. and Langdon, S. (2007) Small and waiting-time behaviour of the thin-film equation. *SIAM J. Appl. Math.* **67** 1776–1807.

[19] Bonnerot, R. and Jamet, P. (1977) Numerical computation of the free boundary for two-dimensional Stefan problem by space-time finite elements. *J. Comput. Phys.* **25** 163–181.

[20] Brackbill, J.U. and Saltzman, J.S. (1982) Adaptive zoning for singular problems in two dimensions. *J. Comput. Phys.* **46** 342-368.

[21] Brackbill, J.U. (1993) An adaptive grid with direction control. *J. Comput. Phys.* **108** 38-50.

[22] Breward, C.J.W., Byrne, H.M. and Lewis, C.E. (2002) The role of cell-cell interactions in a two-phase model for avascular tumour growth. *J. Math. Biol.* **45** 125–152.

[23] Budd, C.J., Leimkuhler, B. and Piggott, M.D. (2001) Scaling invariance and adaptivity. *Appl. Numer. Math.* **39** 261–288.

[24] Budd, C.J. and Piggott, M.D. (2005) The geometric integration of scale-invariant ordinary and partial differential equations. *J. Comput. Appl. Math.* **128** 399–422.

[25] Budd, C.J. and Williams, J.F. (2006) Parabolic Monge-Ampere methods for blow-up problems in several spatial dimensions. *J. Phys. A* **39** 5425–5444.

[26] Budd, C.J. and Williams, J.F. (2008) Moving mesh generation using the Parabolic Monge-Ampere equation. *SIAM J. Sci. Comput.*

[27] Budd, C., Huang, W., and Russell, R.D. (1996) Moving mesh methods for problems with blow-up. *SIAM J. Sci. Stat. Comput.* **17** 305–327.

[28] Budd, C., Huang, W., and Russell, R.D. (2009) Adaptivity with moving grids. *Acta Numer.* 111–241.

[29] Byrne, H.M., King, J.R., McElwain, D.L.S. and Preziosi, L. (2003) A two-phase model of solid tumour growth. *App. Math. Lett.* **16** 567–573.

[30] Byrne, H.M. (1999a) Using mathematics to study solid tumour growth. *Proceedings of the 9th General Meetings of European Women in Mathematics.* 81–107.

[31] Byrne, H.M and Owen, M.R. (2004) A new interpretation of the Keller-Segel model based on multiphase modelling. *Jour. of Math. Biol* **49** 604–626.

[32] Cairncross, R.A., Schunk, P.R., Baer, T.A., Rao, R.R. and Sackinger, P.A. (2000) A finite element method for free surface flows of incompressible fluids in three dimensions. Part I. Boundary fitted mesh motion. *Int. J. Numer.Meth. Fl.* **33** 375-403.

[33] Cao, W., Huang, W. and Russell, R.D. (2002) A moving-mesh method based on the geometric conservation law. *SIAM J. Sci. Comput.* **24** 118-142.

[34] Cao, W., Huang, W. and Russell, R.D. (2003) Approaches for generating moving adaptive meshes: location versus velocity. *Appl. Numer. Math.* **47** 121-138.

[35] Caramana, E.J. and Shashkov, M.J. (1998) Elimination of artificial grid distortion and hourglass type motions by means of Lagrangian subzonal masses and pressures. *J. Comput. Phys.* **142** 521-561.

[36] Carlson, N.N. and Miller, K. (1998) Design and application of a gradient-weighted moving finite element code I: In one dimension. *SIAM J. Sci. Comput.* **19** 728-765.

[37] Carlson, N.N. and Miller, K. (1998) Design and application of a gradient-weighted moving finite element code II: In two dimensions. *SIAM J. Sci. Comput.* **19** 766-798.

[38] Crank. J. and Gupta. R.S. (1972) A moving boundary problem arising from the diffusion of oxygen in absorbing tissue. *J. Inst. Maths. Applics.* **10** 19–33.

[39] Dahmardah, H.O. and Mayers, D.F. (1983) A Fourier-Series solution of the Crank-Gupta equation. *IMA J. Numer. Anal.* **3** 81–85.

[40] DiMilla. P.A., Barbee, K. and Lauffenburger, D.A. (1991) Mathematical model for the effects of adhesion and mechanics on cell migration speed. *Biophys J.* **60** 15–37.

[41] Doedel, E.J. (1981) AUTO, A program for the automatic bifurcation analysis of autonomous systems. *Congr. Numer.* **30** 265–384.

[42] Dorfi, E.A. and Drury, L.O'C. (1987) Simple adaptive grids for 1-D initial value problems. *J. Comput. Phys.* **69** 175–195.

[43] Demirdzic, I. and Peric, M. (1988) Space conservation law in finite volume calculations of fluid flow. *Int. J. Numer. Meth. Fl.* **8** 1037-1050.

[44] Demirdzic, I. and Peric, M. (1990) Finite volume method for prediction of fluid flow in arbitrarily shaped domains with moving boundaries. *Int. J. Numer. Meth. Fl.* **10** 771-790.

[45] Dvinsky, A.S. (1991) Adaptive grid generation from harmonic maps on Riemannian manifolds. *J. Comput. Phys.* **95** 450-476.

[46] Gatenby, R.A. and Gawlinkski, E.T. (1996) A reaction-diffusion model of cancer invasion. *Cancer Res.* **56** 5745-5753.

[47] Gatenby, R.A. and Maini, P.K. (2003) Cancer summed up. *Nature* **421** 321.

[48] Gelinas, R.J., Doss, S.K. and Miller, K. (1981) The moving finite element method: application to general partial differential equations with multiple large gradients. *J. Comput. Phys.* **40** 202–249.

[49] Greenspan, H.P. (1976) On the growth and stability of cell cultures and solid tumours. *J. Theoret. Biol.* **56** 229–242.

[50] Grindrod, P. Ricatti rides again. *Private communication.*

[51] Hansen, E. and Hougaard, P. (1974) On a moving boundary problem with biomechanics. *H. Inst. Maths. Applics.* **13** 385–398.

[52] Harlen, O.G., Rallinson, J.M. and Szabo, P. (1995) A split Lagrangian-Eulerian method for simulating transient viscoelastic flows. *J. Non-Newton. Fluid* **60** 81-104.

[53] Harten, A. and Hyman, J.M. (1983) Self-adjusting grid methods for one-dimensional hyperbolic conservation laws. *J. Comput. Phys.* **50** 235–269.

[54] Heil, M. (2004) An efficient solver for the fully coupled solution of large displacement fluid-structure interaction problems. *Comput. Methods Appl. M.* **193** 1-23.

[55] Huang, W., Ren, Y. and Russell, R.D. (1994) Moving mesh partial differential equations (MMPDEs) based on the equidistribution priciple. *SIAM J. Sci. Comput.* **31** 709-730.

[56] Huang, W. and Russell, R.D. (1996) A moving collocation method for solving time dependent partial differential equations. *Appl. Numer. Math* **34** 1106-1126.

[57] Huang, W. and Russell, R.D. (2011) Adaptive Moving Mesh Methods. *Springer, New York.*

[58] Knupp, P.M. (1995) Mesh generation using vector-fields. *J. Comput. Phys.* **119** 142-148.

[59] Knupp, P.M. (1996) Jacobian-weighted elliptic grid generation. *SIAM J. Sci. Comput.* **17** 1475-1490.

[60] Kuhl, E., Hulshoff, S. and Borst, R.de (2003) An arbitrary Lagrangian Eulerian finite-element approach for fluid-structure interaction phenomena. *Int. J. Numer. Meth. Eng.* **57** 117-142.

[61] Kuraz, M. (2009) An Adaptive Time Discretization to the Numerical Solution of Richards' Equation. *Czech University of Life Science* http://klobouk.fsv.cvut.cz/∼miguel/poster.pdf.

[62] Lacey, A.A., Ockendon, J.R. and Tayler, A.B. (1982) Waiting-time solutions of a nonlinear diffusion equation. *SIAM J. of Appl. Maths.* **42** 1252–1264.

[63] Landman, K.A. and Please, C.P. (2001) Tumour dynamics and necrosis: Surface tension and stability. *IMA J. Math. Appl. Medicine Biol.* **18** 131-158.

[64] Lee, H.K. (2001) For the behaviour of the solutions of a general porous media equations at large time scale. *Information Center for Mathematical Sciences, Seoul University,* **4** 33–88.

[65] Lee, T.E., Baines, M.J., Langdon, S. and Tindall, M.J. (2011) A moving mesh approach for modelling avascular tumour growth. *Math. Comput. Model* Submitted.

[66] Lee, K.A. and Vazquez, J.L. (2003) Geometrical properties of solutions of the Porous Medium Equation for large times. *Indiana University Mathematics Journal* **52** 991–1015.

[67] Liao, G. and Anderson, D. (1992) A new approach to grid generation. *Appl. Anal.* **44** 285-298.

[68] Liao, G. and Xue, J. (2006) Moving meshes by the deformation method. *J. Comput.Appl. Math.* **195** 83-92.

[69] Lubkin, S.R. and Jackson, T. (2002) Mulitphase mechanics of capsule formation in tumours. *J. Biomech. Eng.* **124** 237–243.

[70] Mackenzie, J.A. and Robertson, M.L. (2000) The numerical solution of one-dimensional phase change problems using an adaptive moving mesh method. *J. Comput. Phys.* **161** 537–557.

[71] Mendes, P.A. and Branco, F.A. (1999) Analysis of fluid-structure interaction by an arbitrary Lagrangian-Eulerian finite element formulation. *Int. J. Numer. Meth. Fl.* **30** 897-919.

[72] Miller, K. and Miller, R.N. (1981) Moving finite elements. I. *SIAM J. Numer. Anal.* **18** 1019-1032.

[73] Miller, K. (1981) Moving finite elements. II. *I, SIAM J. Numer. Anal.* **18** 1033-1057.

[74] Muttin, F., Coupez, T., Bellet, M. and Chenot, J.L. (1993) Lagrangian finite-element analysis of time-dependent free-surface flow using an automatic remeshing technique application to metal casting. *Int. J. Numer. Meth. Eng.* **36** 2001-2015.

[75] Osman, K. (2005) Numerical schemes for a non-linear diffusion problem. *MSc Dissertation, University of Reading, UK.*

[76] Parker, J. (2010) An invariant approach to moving-mesh methods for PDEs. *MSc Dissertation, Brasenose College, University of Oxford, UK.*

[77] Pattle, R.E. (1959) Diffusion from an instantaneous point source with a concentration-dependent coefficient. *Q. J. Mech. Appl. Math.* **12** 407–409.

[78] Peterson, R.C., Jimack, P.K. and Kelmanson, M.A. (1999) The solution of two-dimensional free-surface problems using automatic mesh generation. *Int. J. Numer. Meth. Fl.* **31** 937-960.

[79] Piggott, M.D., Gorman, G.J., Pain, C.C., Allison, P.A., Candy, A.S., Martin, B.T. and Wells, M.R. (2008) A new computational framework for multiscale ocean modelling based on adapting unstructured meshes. *Int. J.Numer. Meth. Fl.* **56** 1003-1015.

[80] Please, C.P., Pettet, G.J. and McElwain, D.L.S. (1998) A new approach to modelling the formation of necrotic regions in tumours. *Appl. Math. Lett.* **11** 89–94.

[81] Please, C.P., Pettet, G.J. and McElwain, D.L.S. (1999) Avascular tumour dynamics and necrosis. *Math. Models Methods Appl. Sci.* **9** 569–579.

[82] Ramaswamy, B. and Kawahara, M. (1987) Lagrangian finite-element analysis applied to viscous free-surface fluid flow. *Int. J. Numer. Meth. Fl.* **7** 953-984.

[83] Richards, L. A. (1973) Capillary conduction of liquids through porous mediums. *Physics 1* **5** 318-333.

[84] Roose, T., Chapman, S.J. and Maini, P.K. (2007) Mathematical models of avascular tumour growth. *SIAM Rev.* **49** 179–208.

[85] Russell, R.D., Williams, J.F. and Xu, X. (2007) MOVCOL4: A moving mesh code for fourth-order time-dependent partial differential equations. *SIAM J. Sci. Comput.* **29** 197–220.

[86] Saksono, P.H., Dettmer, W.G. and Peric, D. (2007) An adaptive remeshing strategy for fluid flows with moving boundaries and fluid-structure interaction. *Int. J. Numer. Meth. Eng.* **71** 1009-1050.

[87] Scherer-Abren, G. (2011) Implicit time-stepping for a conservative moving mesh finite difference method. *Private communication.*

[88] Semper, B. and Liao, G. (1995) A moving grid finite element method using grid deformation. *Numer. Meth. Part. D. E.* **11** 603-615.

[89] Soulaimani, A. and Saad, Y. (1998) An arbitrary Lagrangian-Eulerian finite element method for solving three-dimensional free surface flows. *Comput. Method. Appl. M.* **162** 70-106.

[90] Stojsavljeic, J.D. (2007) Investigation of waiting times in non-linear diffusion equations using a moving mesh method. *MSc. dissertation, University of Reading, UK.*

[91] Szabo, P. and Hassager, O. (1995) Simulation of free surfaces in 3-d with the arbitrary Lagrange-Euler method. *Int. J. Numer. Meth. Eng.* **38** 717-734.

[92] Tang, T. (2005) Moving mesh methods for computational fluid dynamics. *Contemporary mathematics AMS* **383** 141–173.

[93] Tenchev, R.T., Mackenzie, J.A., Scanlon, T.J. and Stickland, M.T. (2005) Finite element moving mesh analysis of phase change problems with natural convection. *Int J. Heat Fluid Fl.* **26** 597–612.

[94] Thomas, P.D. and Lombard, C.K. (1979) Geometric conservation law and its application to flow computations on moving grids. *AIAA* **17** 1030–1037.

[95] Thompson, J.F, Warsi, Z.A. and Mastin, C.W. (1985) Numerical grid generation: foundations and applications. *Elsevier North-Holland, Inc. New York, NY, USA.*

[96] Tindall, M.J. and Please, C.P. (2007) Modelling the cell cycle and cell movement in multicellular tumour spheroids. *Bull. Math. Biol.* **69** 1147–1165.

[97] Trulio, J.G. and Trigger, K.R. (1961) Numerical solution of the one-dimensional Lagrangian hydrodynamic equations. *Lawrence Radiation Laboratory Report, University of California.*

[98] Udagawa, N., Fernandez, A., Achilles, E.G., Folkman, R.J. and D'Amato, R.J. (2002) Persistence of microscopic human cancers in mice: Alterations in the angiogenic balance accompanies loss of tumour dormacy. *The FASEB Jour.* **16** 1361–1370.

[99] Vazquez, J.L. (2007) The porous medium equation: Mathematical theory. *Oxford University Press.*

[100] Walkley, M.A., Gaskell, P.H., Jimack, P.K., Kelmanson, M.A. and Summers, J.L.(2005) Finite element simulation of three-dimensional freesurface flow problems with dynamic contact lines. *Int. J. Numer.Meth. Fl.* **47** 1353–1359.

[101] Walkley, M.A., Gaskell, P.H., Jimack, P.K., Kelmanson, M.A. and Summers, J.L. (2005) Finite element simulation of three-dimensional freesurface flow problems. *J. Sci. Comput.* **24** 147-162.

[102] Wang, L.R. and Ikeda, M. (2004) A Lagrangian description of sea ice dynamics using the finite element method. *Ocean Model*, **7**, 21-38.

[103] Ward, J.P. and King, J.R. (1997) Mathematical modelling of avascular tumour growth. *IMA J. Math. Appl. Med. Biol.* **14** 39–69.

[104] Wesseling, P. (2001). Principles of computational fluid dynamics. *Heidelberg: Springer.*

[105] Winslow, A. (1967). Numerical solution of the quasi-linear Poisson equation in a nonuniform triangle mesh. *J. Comput. Phys.* **1** 149-172.