University of Reading

School of Mathematics and Physical Sciences

Final year project

Moving Mesh Methods using monitor functions for the Porous Medium Equation

---

Author:
Theodora Eleftheriou

Supervisors:
Dr. Michael Baines
Dr. Peter Sweby

September 29, 2011

This dissertation is submitted to the Department of Mathematics and Statistics in partial fulfilment of the requirements for the degree of Master of Science

Declaration

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

Signature: .....

Abstract

Many non-linear partial differential equations cannot be solved analytically, and for that reason numerical approximations are required. Thus, an adaptive moving mesh method is used in this dissertation in order to examine the accuracy of the method based on monitor functions.

The purpose of this dissertation is to investigate a moving mesh method, based on a conservation of mass principle applied to the dimensional porous medium equation (PME). Using the evolution of a self-similar solution of the PME, the moving mesh method is investigated in order to determine whether a good representation of these solutions is obtained. It also provides an assessment for the accuracy of the moving mesh method when the choice of monitor function varies. Analytical results for both mass and arc-length monitor functions are considered and a brief evaluation of their accuracy is produced.

Acknowledgments

First of all I would like to thank my supervisors, Mike Baines and Pete Sweby for their help and guidance through this study. I am very grateful for their continuing patience in answering all my questions and always being enthusiastic and supporting. It was really pleasant to work with them.

Also a big thanks to all of my friends for their help and support they have given me during the endless nights.

Last but not least, the most important gratitude is for my family for their love, support and their encouragement to never give up and to pursue my goals.

Contents

List of Figures

Chapter 1

Introduction

Many differential equations are too complicated to derive analytical solutions. For that reason over the last decades many studies have been done in order to find numerical approaches to those situations. In particular adaptive mesh methods have been developed aiming to give a good representation of the solution of partial differential equations; (PDEs). Often the solution of these PDEs involves large solution variations such as, at shock waves and steep fronts, thus adaptive methods have been applied to a variety of problems that cannot be solved exactly, particularly non-linear differential equations.

Adaptive mesh methods become preferable to a fixed mesh scheme when these areas of interest represent only a fraction of the domain being investigated. Consequently, using these adaptive techniques is computationally efficient since the resolution is concentrated on selected regions and we avoid the refinement of the entire domain. As a result this will efficiently keep the computational costs down and save time, especially when we are interested in higher dimensions.

There are three main types of adaptive mesh methods, namely h-refinement, p-refinement and r-refinement.

The h-refinement approach is the most usual method where extra nodes are added or removed from the existing mesh. Using an error indicator for example, it is possible to identify where the solution has insufficient regularity in that area and improve the accuracy of the solution by introducing

more computational nodes on the mesh. This might lead to local refinement or coarsening of the mesh. The p-refinement technique increases the order of the polynomials used in finite element approximations, seeking a better and more accurate approximation solution in each element. Finally, r-refinement is the least common method which describes the change in the position of the existing computational nodes assuming that the total number of nodes remains fixed in time. The scope of this technique is to enhance the overall productivity of the existing numerical approximation. All these adaptive techniques have been applied on existing numerical methods by several authors in their attempt to achieve a more accurate and more robust representation of the numerical solution.

In this project we will concentrate on an application of the r-refinement method where r stands for relocation. This technique is also well-known as the *moving mesh method*. It aims to move the mesh to the regions where needed. This method is normally used for the numerical approximation of time-dependent problems. The reason for that is due to the mobility of the mesh, consequently time integrators are easier to be taken into consideration. The r-refinement method has some advantages over the other two methods since it is a flexible and easy method to implement. No mesh points are added or removed and by the time that the program has been set up, the structure of the method remains consistent. On the other hand, the main limitation of this method is the difficulty of timestepping, since the mesh nodes are changing positions all the time and may tangle.

This dissertation will explore this r-refinement method by generating numerical results for a specific computational example, namely the porous medium equation **(PME)**. The investigation focuses on a particular velocity-based moving mesh method, considering different choices of monitor functions applied within the method. This will eventually lead to a wider evaluation for the accuracy and the reliability of this technique.

Although the porous medium equation is in the category of those PDEs that have complex behaviour and difficulties in predicting their analytical solutions, it has a particular exact self-similar solution in a special (self-similar) case. Using this equation as a computational example gives us the

opportunity to check the potential of this adaptive method to represent self-similar solutions of the given problem.

In Chapter Two we begin by studying equidistribution, which is a standard way of distributing mesh points and moving them. The equidistribution principle is applied within the computational cells according to the choice of the monitor function that has been taken. We provide the three most commonly used monitor functions and we show graphically how the equidistribution of a uniform mesh is presented when different monitor functions are applied. Further investigations on the choice of the monitor function are considered in Chapter Three and some examples of the applications of the monitor functions are provided.

In Chapter Four we introduce the advantages of some of the properties of the PME, and how these properties are applied in our investigations. The property of scale invariance according to non-linear diffusion equations is discussed and also how the similar solutions are derived. A special case of similar solutions has been applied for the purposes of this dissertation known as 'self-similar' solutions. Self-similar solution has been used as an initial condition and implemented in the moving mesh algorithm aiming to present the effectiveness of this scheme.

In Chapter Five we introduce the velocity-based method, and how it is applied. A full algorithm to determine the mesh velocity is given as well. Chapter Six makes an introduction of the *'location-based methods'*, and gives a brief description of the applications of MMPDEs.

Then, in Chapter Seven an example of a moving mesh algorithm using mass monitor function has been constructed and extensively investigated with direct reference to PME with $n = 2$, and numerical results are provided. In Chapter Eight we describe the velocity-based method when an arc-length monitor function is chosen.

Finally in the last chapter, Chapter Nine, a summary is presented and some conclusions for this work are stated. The limitations of the adaptive moving mesh method are provided and suggestions for possible improvements as a possible future avenue of study are explored.

Chapter 2

The Equidistribution principle

Mesh generation and adaptive methods play a crucial role in the solution of both ordinary and partial differential equations.

Adaptive mesh methods typically construct an irregular mesh which is generated by a transformation from a computational domain to the physical domain where the given problem is being solved.

2.1 Introduction

The equidistribution principle plays a fundamental role in the moving mesh method. The idea was first introduced by de Boor [5] to solve boundary value problems for ordinary differential equations (ODEs). Referring back to de Boor's theory, the equidistribution principle is critical if we wish to generate a discrete approximation of a function on a non-uniform mesh. The main concept underlying this method is to equally distribute the volume under the integral of some quantity, in each computational cell of the mesh. This quantity is defined by the user and is called a *'monitor function'*. One possible use of the monitor function is to therefore equdistribute the solution error of the taken mesh points, over each subinterval.

In [1], Dorfi and Drury adapted the 'equidistribution principle' to generate a moving finite difference method for solving 1D initial value problems (IVPs).

The equidistribution principle is mainly used for grid relocation in one spatial dimension, for which a monitor function $M(x)$ is introduced. To be more precise, usually the monitor function is a positive function of a function $u(x)$ and its derivatives.

For the purposes of this dissertation we only consider the equidistribution principle in one spatial dimension.

Without loss of generality we consider a computational domain and a physical domain, denoted by Ω_c and Ω respectively. The domains both have a unit interval [0,1] in space, thus we can write $x, \xi \in [0,1]$. We seek a mapping $x(\xi, t)$ from ξ to x at time t. We assume boundary conditions $x(0,t) = 0$ and $x(1,t) = 1$.

According to White [2], the continuous form of the equidistribution principle is given as

$$\int_0^{x(\xi,t)} M(x(\xi,t)dx = \xi \int_0^1 M(x(\xi,t),t)dx \tag{2.1}$$

This integral form has been used later by various mathematicians using different techniques to try to give a good approximate solution of a given partial differential equation on an equidistributed mesh. This will be explained in the next chapter in more detail. However, a brief discussion will be given now, in order to give a clearer idea of how we derive the 'equidistribution principle'.

Huang, Ren and Russell [3] used the equidistribution principle (2.1) and differentiated it with respect to ξ. This leads to

$$M(x(\xi,t),t)\frac{\partial}{\partial \xi}x(\xi,t) = \theta(t) \tag{2.2}$$

where $\theta(t) = \int_0^1 M(x(\xi,t),t)dx$

White solved numerically the time-independent version (2.2) in an effort to generate an adaptive mesh which would represent the approximate solution of two boundary-value problems.

In the case where we differentiate the equidistribution principle (2.1) twice

with respect to ξ, leads to

$$\frac{\partial}{\partial \xi}\left(M(x(\xi,t),t)\frac{\partial}{\partial \xi}x(\xi,t)\right) = 0 \tag{2.3}$$

which are considered in the area of *Quasi-static equidistribution principles.*

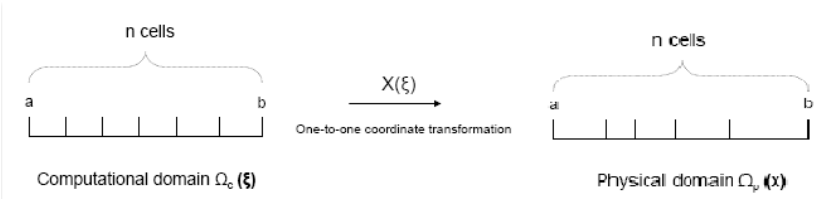
Continuing from (2.3), Baines [4] solved again numerically the time-independent version of this equation by applying an iterative algorithm technique. In order to solve this equation, a discretisation has been used which leads to a *tridiagonal system.*

Based on this method, we subsequently equidistribute the mesh according to several monitor functions and represent the function on an equally spaced mesh in Ω_c. We will also study the efficiency of each monitor function, and evaluate the most appropriate mesh for the representation of our solution.

As the result of the mapping, the grid points x_i are irregularly spaced in the physical space $\alpha \leq x_i \leq b$ and are related to the regularly spaced gridpoints ξ_i in the computational space $0 \leq \xi_i \leq 1$ by discrete values of the continuous variable

$$\xi = \frac{\int_\alpha^x M(\widetilde{x})d\widetilde{x}}{\int_\alpha^b M(\widetilde{x})d\widetilde{x}}. \tag{2.4}$$

The variable ξ can be defined as a ratio which actually describes the reverse mapping that we attempt to achieve from the computational to the physical domain.



Differentiating twice equation (2.4) with respect to ξ , gives the differential equation

$$(M(x)x_\xi)_\xi = 0$$

as in (2.3).

The M however will generally depend on x (through the function), leading as a result, the equation above to be nonlinear. However, the differential equation can be solved using the following iterative algorithm (functional iteration)

$$(M(x^p)x_\xi^{p+1})_\xi = 0 \qquad (p = 0, 1, ...) \tag{2.5}$$

where x^0 is an initial guess that can be applied for the starting point on a uniform grid, provided that it converges. Applying the boundary conditions $x(0) = 0, x(1) = 1$, to the solution of the equation, it then provides a distribution $x(\xi)$ for a given monitor function M. The number of iterations used in order to solve the problem depends on a given tolerance.

Computationally this method will produce a mesh for the function and then using it we can update the mesh by transferring the function from the old mesh to the new one.

In practice the iteration (2.5) can be approximated using the *discretization method*.

$$M(x_{j+\frac{1}{2}})(x_{j+1} - x_j) = M(x_{j-\frac{1}{2}})(x_j - x_{j-1})$$

for $0 < i < i_{max}$, so that

$$M(x^p_{j+\frac{1}{2}})(x^{p+1}_{j+1} - x^{p+1}_j) - M(x^p_{j-\frac{1}{2}})(x^{p+1}_j - x^{p+1}_{j-1}) = 0$$

This leads to the matrix system

$$T(x^p)x^{p+1} = \mathbf{b}$$

where T is a *tridiagonal matrix* and $\mathbf{b}$ comes from overwriting the boundary conditions $x_0 = 0$ and $x_1 = 1$.

In order now to solve this matrix algorithm we apply the direct method of the *'Thomas Algorithm'*.

Additionally for evolving the iteration method we determine a preset stopping tolerance. Computationally we use a 'do loop' which stops when the error is smaller than the tolerance. Consequently, we accumulate a sequence of approximations that converges sufficiently to the solution. More precisely, we set a figure for the tolerance and then check the total error over all the output points. If the error is smaller than the set tolerance, the program automatically saves that gridpoint value, otherwise the program keeps running the 'do loop' to the next output point.

The aim therefore, of applying this system is to reach a sufficient level of convergence.

The monitor function is usually chosen according to the given data function. This implies that we may have different types of monitor functions. The choice therefore, of the monitor function depends on the given function. At this point we will explain three possible applications of 'equidistribution principle' and the corresponding monitor functions.

2.2 Choices of Monitor functions

2.2.1 Gradient Monitor function

In the first case, the data is a continuous function $f(x)$ and a possible monitor function is:

$$M(x) = \frac{df}{dx}$$

which is called the *gradient monitor function* since

$$\int_{x_{i-1}}^{x_i} \frac{df}{dx} dx = f_i - f_{i-1}.$$

Equal intervals on the f axis are taken, and the corresponding values on x-axis are obtained.

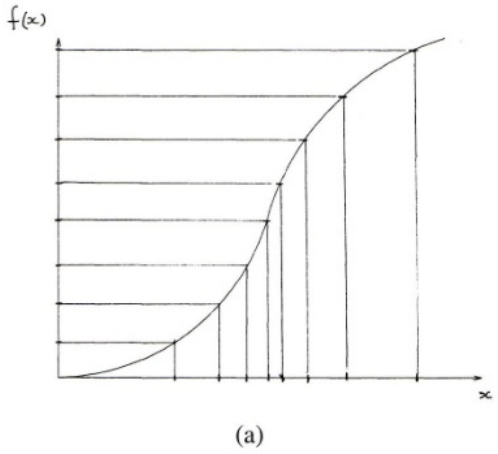


Figure 2.1: Equidistribution of function f

The graph illustrates that more points are concentrated below the steep front. However, as a result this might lead to some inefficiencies since the points give more information below the steep gradient and not enough resolution in the rest of the region.

2.2.2 Arc-Length Monitor function

Alternatively, we can use the arc-length monitor function

$$M(x) = \frac{ds}{dx}$$

where

$$s(x) = \int \left(1 + \left(\frac{df}{dx}\right)^2\right)^{\frac{1}{2}} dx$$

which equidistributes the arc-length s.

Graphically we equally divide the total arc length and mark the corresponding x values on the x-axis.

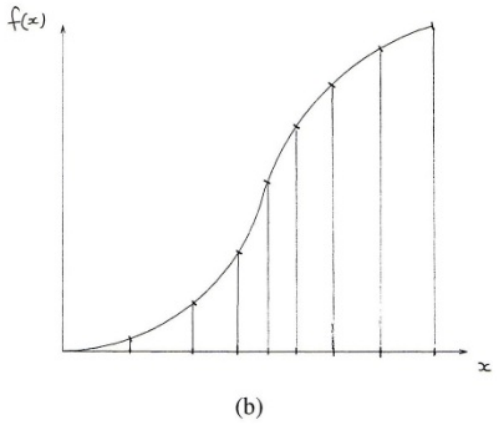


Figure 2.2: Equidistribution with arc-length monitor function

The points now, as we can observe from the above graph, still concentrate in the steep region but are more evenly spaced, compared with Figure (2.1).

2.2.3 Mass Monitor function

The mass monitor function has the form of

$$M(x) = f(x)$$

and by applying this monitor function we equidistribute the area under the function in equal parts. Using this monitor function we eventually aim to produce a scheme which conserves partial masses of a time dependent solution for all time.

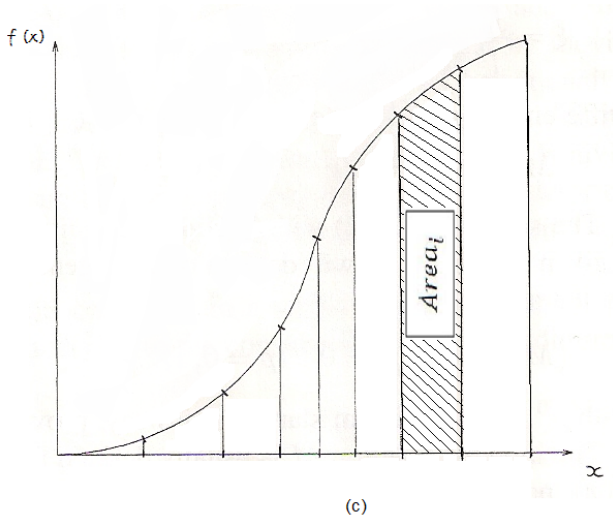


Figure 2.3: Equidistributed mesh using a mass monitor function

The unique feature of the mass monitor function is that using this monitor it keeps the area of each section constant while the time changes. The importance of the monitor function in the adaptive moving mesh method will be discussed thoroughly in the chapter that follows.

2.3 Numerical Results

In this section, we will start our investigation by constructing a program with an initial test function, in order to check and make more obvious the applications of the 'equidistribution principle' based on different choices of monitor functions.

2.3.1 Numerical Results for Equidistributed meshes

For the purposes of this dissertation we consider the function

$$u(x) = \left(1 - \frac{x^2}{4}\right)^{\frac{1}{2}} \tag{2.6}$$

defined on the interval $x \in [0, 2]$. This function is chosen since it will be the initial data for our later investigations and has an infinite slope at $x = 2$.

Starting our investigation, we will represent this function discretely on a uniform mesh with a finite number of mesh points given as

$$0 \equiv x_0 < x_1 < \ldots < x_{N-1} < x_N \equiv 2$$

Considering the equidistribution principle (2.1) we seek an equidistributed mesh that represents the function. The mesh consists of gridpoints x_i for $i = 1, \ldots N$. The grid points will be generated in such a way that the equidistribution principle will be satisfied on a chosen monitor function.

Using now the Baines algorithm which can be found in [4], we constructed a FORTRAN program in order to check the equidistribution principle that has been applied on the given function.

The numerical results presented here are the numerical solutions which have been obtained after starting the equidistribution principle algorithm (iteration) on a uniform mesh using the three monitor functions mentioned above. From this outcome we can compare the efficiency of the chosen monitor function applied. An evaluation of the significance of the chosen monitor function of the previous section, will be given further on.

We used a total number of computational nodes of $N = 20$ and an initial space size of $dx = 0.1$. In order to solve the tridiagonal system we applied the Thomas Algorithm to the iterative method to calculate the position of the equidistributed grid points. Initially we start the algorithm by equally spacing the existing mesh nodes and then to obtain the new updated solutions, we update the solutions' values. To achieve that, we first to update the position of the nodes. The boundary conditions that we imposed for the mapping are $x = 0$ at $\xi = 0$ and $x = 2$ at $\xi = 1$.

The Figure (2.4) shows the output after applying the 'mass monitor function' on the test function. As we can observe from Figure (2.4), the mesh points are concentrated more densely on the top whereas at the bottom of the line no points are formed, which might cause inefficiencies.

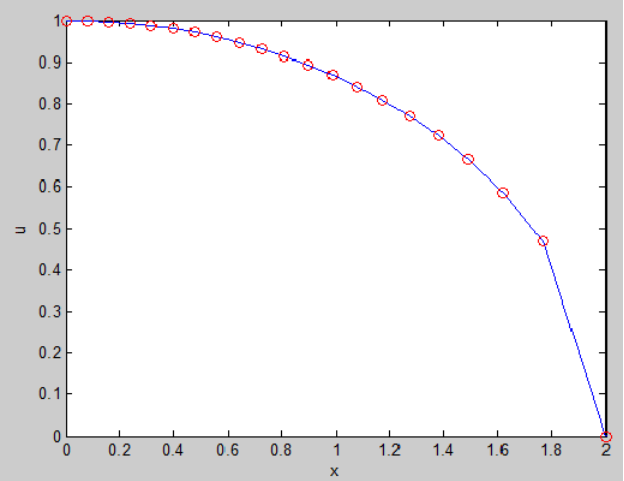


Figure 2.4: Equidistributed mesh using a 'mass monitor function'

Applying now the 'arc-length monitor function' the points are equidistributed over the arc length as it is shown in Figure (2.5). In comparison with the previous approach, it is obvious that leads to a smoother representation of the solution when the arc-length monitor function is applied.

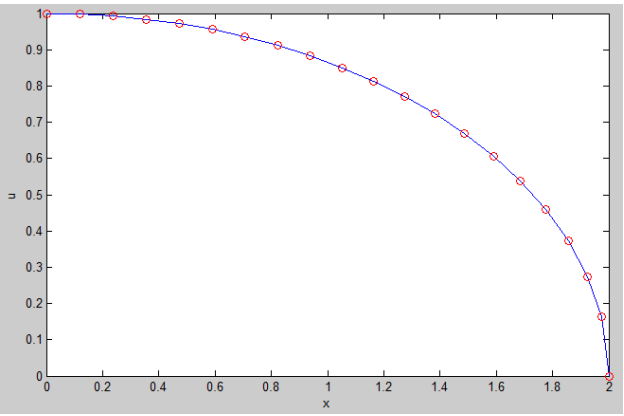


Figure 2.5: Equidistributed mesh using an 'arc-length monitor function'

The arc length monitor function is one of the most commonly used functions applied. The reason for that is because the arc length monitor function

causes the mesh points to cluster near where steep fronts occur but also keeps the points in 'flat' regions. Therefore, since more mesh points are gathered in the areas of large variations, we have better representation of the behaviour of the solution.

Figure (2.6) describes the behaviour of the 'equidistribution principle' after applying the *gradient monitor function*. The points are clustered more at the bottom now, using this monitor function. However, we do not use this monitor function further in this dissertation. Our calculations will be according to the moving mesh method using a *mass and an arc-length monitor functions*.

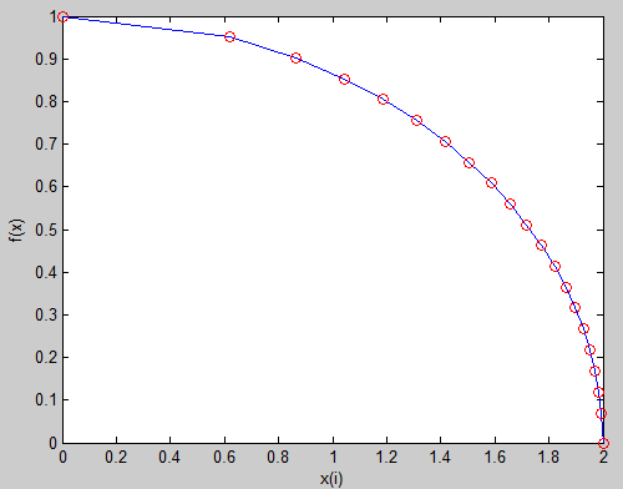


Figure 2.6: Equidistributed mesh using a 'gradient monitor function'

Chapter 3

Monitor Functions for Moving Mesh Solutions of PDEs

3.1 Introduction

We shall apply monitor functions to moving mesh solutions of PDEs. The choice of monitor function plays a fundamental role in the adaptive moving mesh methods. Consequently, we have to be precise with the choice of monitor function when it is applied to numerical approximation schemes. An inappropriate choice of monitor function might lead to severe inefficiencies and affect the optimisation of the initial data.

3.1.1 Applications of Monitor Functions to PDEs

Most moving mesh methods for PDEs use monitor functions. However, in order to give a clear idea of the use of the monitor function we have to refer back to the equidistribution principle and see how these two are connected. Equidistibution is the description of node placement so that a quantity is equally distributed. When the method is applied to a non-linear diffusion equation the quantity mentioned might be area. If we need to examine the steep fronts then we might choose to equidistribute the arc length.

In particular, we shall be applying the idea of monitor functions to the porous medium equation which can model a gas "bubble" spreading in a

porous medium.

The main role of the monitor function is to accentuate the importance of the various parts of the domain by assigning a 'weight-value' to each location.

To make the adaptive moving mesh method more reliable, further studies have been done by Huang and Cao et al, in [7], describing analytically the strategy of choosing the most suitable monitor function. According to Budd, Huang and Russell [6] the monitor function is defined to be always a non-negative function.

Types of monitor functions that are commonly used are:

- Arc length;
- Combination of gradient and curvature;
- Truncation error or solution residual.

In the previous section we have already used the monitor functions for equidistribution. For further investigations in this dissertation we want to use the same monitor functions in order to move the mesh, using a moving mesh method. However, we first describe the Porous Medium Equation which we shall use for numerical experiments. We then discuss two approaches to constructing moving meshes, *velocity-based method* and *location-based method*.

Chapter 4

The Porous Medium Equation (PME)

4.1 Introduction

Based on Darcy's extensive investigations [16], much work has been done in modelling the flow and heat transfer through porous media. From this work it is possible to study a broad range of different fields and applications such as ground-water hydrology, chemical reactors, petroleum reservoir and geothermal operations.

The PME is a second order non linear diffusion equation which has many applications. For example, in biological modelling PME describes common phenomena for tissues of the human body. Bone cartilage and muscle are defined as porous media and their functioning depends on the flow of fluids through them. The PME helps to understand the pathological conditions related to these materials.

The study of flow and heat transfer is usually based on the transport equations resulting from differential balance laws. To predict global effects such as flow resistance or heat flux in a given situation requires detailed information of the surrounding velocity and temperature fields. This information is extracted from the solution of the associated transport equations, subject to the pertinent boundary conditions. When flow through a complex struc-

ture such as a porous medium is involved, these equations are still valid inside of the pores, but the geometric complexity prevents general solutions of the detailed velocity and temperature fields. Instead, some form of 'macroscopic' balance equations based on the average over a small volumetric element must be employed. A common practice is to replace the 'microscopic' momentum and energy equations by the corresponding 'macroscopic' equations with the help of some well-established empirical relations [17].

Porous flows, illustrated by the diffusion of gas through a porous medium and taking into consideration the conjecture of Darcy's law relating the velocity to the pressure gradient, give rise to the porous medium equation.

The variables of density (u), pressure (p), and velocity (v) are the parameters that distinguish the flow of the gas.

The assumption that the frictional drag is directly proportional to velocity for low speed flow, determines the pressure drop in the flow in the porous medium. This leads to Darcy's law which relates the pressure drop and velocity in an unbounded porous medium. Consequently, Darcy's law is an empirical law for dynamics of the flow through a porous medium.

4.2 Derivation of PME from Darcy's law

It is assumed that the gas obeys the conservation of mass equation

$$\rho u_t + \nabla \cdot (u\mathbf{v}) = 0 \tag{4.1}$$

where ρ is the constant porosity of the medium and $\mathbf{v}$ is given by Darcy's law, which is stated as

$$\mu\mathbf{v} = -\kappa\nabla p \tag{4.2}$$

where $\mu \geq 0$ is the viscosity of the gas and κ is the permeability of the medium. Both quantities are assumed to be constant parameters.

Furthermore, since the gas is assumed to be ideal, the pressure and density are related by

$$p = p_0 u^\gamma \tag{4.3}$$

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 both equations (4.2) and (4.3) into the conservation of mass equation (4.1) leads to the equation

$$u_t = c\nabla \cdot (u^\gamma \nabla u)$$

where c is constant and is given as

$$c = \frac{\kappa p_0 \gamma}{\mu \rho}.$$

The constant c can be scaled out of the problem and if this is done and also if we equate γ with n then we obtain the original Porous Medium Equation in the form

$$u_t = \nabla \cdot (u^n \nabla u) \tag{4.4}$$

where it is assumed that n is always positive, $n > 0$. Usually the boundary condition applied on the porous medium equation is $u = 0$ on all boundaries.

There has been an extensive analysis and detailed explanation on the properties of the PME. The most comprehensive analysis and theory can be found in [14, 15, 16].

4.3 Properties of the one dimensional PME

The PME has several properties that we will use in our study, which we need before obtaining the numerical solution. Studies based on the properties of the PME, can be found in [18]. Explanations are given in detail, and it also includes all proofs of the properties as well.

The most well-established properties of the PME are the conservation of mass and stationary centre of mass. However, for the purposes of this dissertation we are interested only in conservation of mass.

4.3.1 Proof that PME conserves mass

In this section we will provide a proof that the one dimensional PME conserves mass in time. For this purpose we show that the derivative of the total mass in time is zero.

In one dimension the PME is

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(u^n \frac{\partial u}{\partial x}\right) \tag{4.5}$$

with boundary conditions $u = 0$ at the boundaries, $x_0(t)$ and $x_n(t)$, say.

Proof. Define mass in $(x_i(t), x_n(t))$ as

$$\int_{x_0(t)}^{x_n(t)} u(x,t) - */41dx \tag{4.6}$$

Differentiating (4.6) and substituting $\frac{\partial u}{\partial t}$ from (4.5) leads to

$$\begin{aligned}
\frac{d}{dt}\int_{x_0(t)}^{x_n(t)} u dx &= \int_{x_0(t)}^{x_n(t)} \frac{\partial u}{\partial t} dx + \frac{dx_n(t)}{dt}\frac{d}{dx_n(t)}\int_{x_0(t)}^{x_n(t)} u(x_n(t))dx \\
&+ \frac{dx_0(t)}{dt}\frac{d}{dx_0(t)}\int_{x_0}^{x_n} u dx \\
&= \int_{x_0(t)}^{x_n(t)} \frac{\partial u}{\partial t} dx + \dot{x}_n(t)u(x_n(t)) - \dot{x}_0(t)u(x_0(t)) \\
&= \int_{x_0(t)}^{x_n(t)} \frac{\partial}{\partial x}\left(u^n \frac{\partial u}{\partial x}\right) dx + [\dot{x}u]_{x_0(t)}^{x_n(t)} \\
&= \left[u^n \frac{\partial u}{\partial x} + \dot{x}u\right]_{x_0(t)}^{x_n(t)} \\
&= 0 \qquad (4.7)
\end{aligned}$$

using the boundary condition $u = 0$. Hence,

$$\frac{d}{dt}\int_{x_0(t)}^{x_n(t)*9} u(t) = 0 \qquad (4.8)$$

giving finally

$$\int_{x_0(t)}^{x_n(t)} u(t) = 0, \qquad (4.9)$$

□

as required.

Using this proof we can state that the PME satisfies conservation of mass over the whole domain.

4.3.2 Scale Invariance and Self-Similar solutions

It is widely accepted that the primary aim of applying an adaptive method to a porous medium equation is to give an accurate representation of the solution. For that reason when any numerical approximation is applied to solve

the non-linear diffusion problem, it is expected that the numerical solution will approximate an exact solution.

For the PME an exact solution is a similarity solution which will be used to validate our numerical work. Using the similarity solutions we will confirm the validity of the numerical results that we obtain, comparing them with the exact solution of the problem.

Self-similar solutions, used for time-dependent problems involving spatial distributions of its variables at different times, can be obtained by a similarity transform which is a transformation that maintains certain features of a function or curve. A specific example of a similarity transformation is a scale-invariant transformation. However, a necessary condition needs to be satisfied. In particular, in order to derive a self-similar solution, it is essential that the PDE must be scale-invariant.

Therefore, in order to get a clear idea of what similarity solutions represent and how they can be derived, firstly we need to ascertain what the scale invariance is, and how this is involved in our investigation.

4.3.3 Scale Invariance

In order to derive the similarity solution, firstly we need to obtain the scale-invariant transformations of the PME. When applying a scaling transformation all the variables are scaled by powers of a common factor λ.

Given a set of values of (u, x, t) which satisfy the PDE

$$u_t = f(x, u, u_x, u_{xx}, \ldots)$$

under consideration, an invariant transformation of (x, u, t) to a new system $(\bar{x}, \bar{u}, \bar{t})$ leaves the PDE unchanged. Subsequently the equation that is satisfied by (x, u, t) is satisfied by $(\bar{x}, \bar{u}, \bar{t})$.

A prerequisite to obtain the similarity solutions for the PDE is a subclass of invariances known as scaling transformation given as

$$x = \lambda^{\beta}\bar{x}, \qquad u = \lambda^{\gamma}\bar{u}, \qquad t = \lambda\bar{t} \tag{4.10}$$

where β and γ are constants to be found, and λ is an arbitrary positive scaling quantity [19]. If the PDE remains unchanged after the mapping then it is said to be scale invariant.

Given the PME

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(u^n \frac{\partial u}{\partial x}\right) \tag{4.11}$$

then by applying the transformation (4.10), the left hand side of the (4.11) can be written as

$$\frac{\partial u}{\partial t} = \frac{\partial(\lambda^\gamma \bar{u})}{\partial(\lambda \bar{t})} = \lambda^{\gamma-1}\frac{\partial \bar{u}}{\partial \bar{t}}. \tag{4.12}$$

Furthermore using the same transformation applied to the right hand side of (4.11) gives

$$\begin{aligned} \frac{\partial}{\partial x}\left(u^n \frac{\partial u}{\partial x}\right) &= \frac{\partial}{\partial(\lambda^\beta \bar{x})}\left(\lambda^{n\gamma}\bar{u_n}\frac{\partial(\lambda^\gamma \bar{u})}{\partial(\lambda^\beta \bar{x})}\right) \\ &= \lambda^{(\gamma(n+1)-2\beta)}\frac{\partial}{\partial \bar{x}}\left(\bar{u_n}\frac{\partial \bar{u}}{\partial \bar{x}}\right). \end{aligned} \tag{4.13}$$

To derive the transformed general PDE we need to equate (4.12) with (4.13)

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

Finally, in order for the given PDE (4.11) to be scale invariant under the transformation (4.10) we require

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

$$2\beta - n\gamma = 1 \tag{4.15}$$

giving a class of symmetries. For evaluating the constants β and γ uniquely, another relation is required.

As already shown, the PME with zero boundary conditions satisfies

$$\int_{x_o(t)}^{x_n(t)} u(t)dx = \text{const. in time.}$$

Then under the scaling transformation (4.10) becomes

$$\int_{x_o(t)}^{x_n(t)} u(t)dx = \int_{x_o(t)}^{x_n(t)} \lambda^\gamma \bar{u} d(\lambda^\beta \bar{x}) = \text{constant in time.}$$

Equating the λ terms on each side we obtain

$$\lambda^{\gamma+\beta} = \lambda^0 \tag{4.16}$$

$$\text{thus} \qquad \gamma + \beta = 0. \tag{4.17}$$

Together with (4.14), equation (4.17) leads to the conclusion that (4.11) with mass conserved (or zero boundary conditions) is scale-invariant. Solving (4.15) and (4.17) simultaneously,

$$\beta = \frac{1}{n+2} \qquad \gamma = \frac{-1}{n+2}. \tag{4.18}$$

In particular, when $n = 2$ then $\beta = \frac{1}{4}$ and $\gamma = -\frac{1}{4}$.

To obtain the self similar solution of the PDE set the scale-invariant similarity variables $\frac{u}{t^\gamma}$ and $\frac{x}{t^\beta}$ to be related by

$$\frac{u}{t^\gamma} = f\left(\frac{x}{t^\beta}\right) \tag{4.19}$$

where f is a function to be determined. For the PME (4.11) with $n = 2$, the function is

$$f = \left(1 - \frac{1}{4}\left(\frac{x}{t^\beta}\right)^2\right)^{\frac{1}{2}} \tag{4.20}$$

with $\beta = \frac{1}{4}$ so that

$$u = \frac{1}{t^{\frac{1}{4}}}\left(1 - \frac{1}{4}\left(\frac{x}{t^{\frac{1}{4}}}\right)^2\right)^{\frac{1}{2}} \tag{4.21}$$

is the self-similar solution.

To summarise, using the scaling transformations that we explained in the previous section, a similarity solution for the PME is obtained. This solution is a solution of the PME which is invariant. Rescaling the particular parameters (x, u, t), using (4.10), leads to the similarity solution which still satisfies the PDE.

Below we present a graph of a set of self-similarity solutions, as found in [20], at three different times. This figure shows the similarity solutions in one spatial dimension, describing the behaviour of the self-similar solution of the PME.

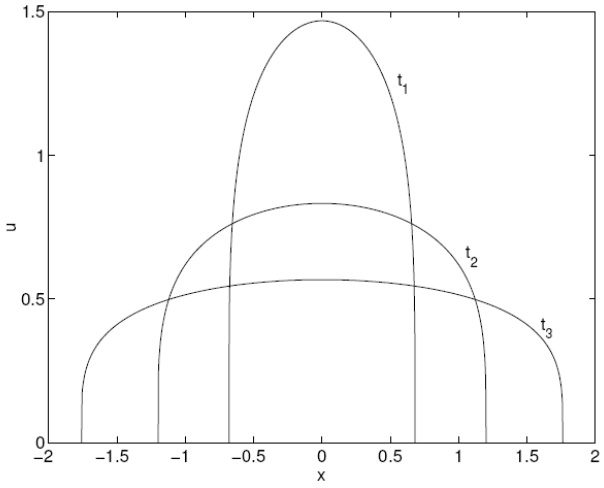


Figure 4.1: Similar solutions of the porous medium equation when $n = 2$

Assuming that in one dimension PME the solution $u(x, t)$ is symmetrical about its centre of mass, then using the condition of symmetry enables us to make our investigations using only the half region $x(t) \in [0, b(t)]$. Then since we are working on the half region, the boundary conditions are given as

$$\begin{aligned} \frac{\partial u}{\partial x} &= 0 \qquad \text{at } x = 0 \\ u &= 0 \qquad \text{at } x = b. \end{aligned}$$

In this project we will construct numerical solutions of the PME using moving mesh meshes. We have seen that there exists a self-similar solution of the porous medium equation, meaning that we can check our implementation of the numerical approximation to calculate the numerical solution against an exact solution.

Applying the moving mesh method to the porous medium equation, gives us the opportunity to investigate whether this method can give a useful representation of the solution. This enables us to compare our computations against this known solution and carry out a critical evaluation for the efficiency of using an adaptive moving mesh method.

The general form of the porous medium equation in one-dimension is

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

where n is an integer greater than zero.The porous medium equation that we used for the purposes of this dissertation in the case where $n = 2$ is

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(u^2 \frac{\partial u}{\partial x}\right), \tag{4.22}$$

with an initial condition (2.6) taken from the self-similar solution and zero boundary conditions on the moving boundaries.

This problem has the similarity solution

$$u(x,t) = \frac{1}{t^{\frac{1}{4}}}\left[1 - \frac{1}{4}\left(\frac{x^2}{t^{\frac{1}{4}}}\right)\right]^{\frac{1}{2}}. \tag{4.23}$$

and the initial function corresponds to time $t = 1.0$.

Chapter 5

A Velocity-Based method

5.1 Velocity-Based method

Velocity based methods belong to the family of 'Arbitrary Lagrangian-Eulerian methods'(ALE). This method applies a Lagrangian moving co-ordinate system, providing a mesh velocity v. Each node is assigned a velocity by which it moves. Applying time integration to the mesh velocity enables us to find the mesh point locations.

A conservation-based method is a very natural technique, since it is a standard approach in fluids and similarity solutions are convected with the velocity [21]. For the conservation-based method it is necessary to have conservation of mass over the whole region $[0, x_N]$.

5.1.1 Calculating the mesh velocity

In the conservation-based method in order to generate the velocity we take the area of any section to be constant in time,

$$\int u dx = \text{constant time} \tag{5.1}$$

for any interval. This corresponds to the monitor u, which is called the *mass monitor*.

Differentiate both sides with respect to t, giving

$$\frac{d}{dt}\int udx = 0. \tag{5.2}$$

Then, using the Leibnitz Rule we obtain

$$\frac{d}{dt}\int_0^x udx = \int_0^x u_t dx + [uv]_0^x$$

which can be simplified to

$$\int_0^x u_t dx + uv = 0 \tag{5.3}$$

since $v = 0$ at $x = 0$.

Inserting the porous medium equation (4.22) into equation (5.3) we get

$$\int_0^x (u^2 u_x)_x dx + uv = 0 \tag{5.4}$$

and carrying out the integral leads to

$$u^2 u_x + uv = 0$$

since u_x is zero at $x = 0$ by symmetry (see Figure (4.1)). Finally we use this equation, rearranged, so that we get the velocity as

$$v = -uu_x = -\frac{1}{2}(u^2)_x \tag{5.5}$$

provided that $u \neq 0$.

The method approximates (5.5) by

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

and uses v to compute the x_i values and u_i values in time. Updating the mesh points, firstly we calculate the new x values from the velocity, using

the Explicit Euler method,

$$x_i^{n+1} = x_i^n + \Delta t \cdot v_i^n \tag{5.7}$$

where $v_i = \frac{dx_i}{dt}$ is the velocity at the point x_i and is calculated from (5.5), Δt is the time step and x_i^{n+1} is the position of x_i at the next time step. The velocity that we obtain is such that partial masses have being conserved.

5.1.2 How to evaluate the u values

Since we know that the area values of the integral for u^{n+1} are constant in time by (5.1) we may write

$$\int_{x_{i-1}t_{(n+1)}}^{x_i t_{(n+1)}} u dx = \int_{x_{i+1}(1)}^{x_{i-1}(1)} u dx = c_i, say \tag{5.8}$$

where $t = 1$ is the initial time.

Using now the mid-point rule we get the approximation

$$(x_{i+1} - x_{i-1})u_i = c_i \tag{5.9}$$

and finally

$$u_i^{n+1} = \frac{c_i}{x_{i+1}(t_{n+1}) - x_{i-1}(t^{n+1})} \tag{5.10}$$

which can be used in the numerical approximation to find the new values of u_i at the next time step.

To make the procedure more obvious and easier to understand, we give the algorithm on how to update the solution of PME.

Algorithm:

Given a mesh with meshpoints x_i^n and values u_i^n ; $i = 0, \ldots, N$, at $t = 1$

1. Compute the mesh velocity from (5.10)

$$v_i^n = -\frac{1}{2} \cdot \frac{(u_{i+1}^n)^2 - (u_i^n)^2}{x_{i+1}^n - x_i^n}$$

   in each interval and use linear interpolation to give its value at x_i

2. Using the Forward Explicit Euler scheme, compute the updated mesh x_i^{n+1}

3. Compute the updated solution of u_i^{n+1} from (5.7) by evaluating

$$u_i^{n+1} = \frac{c_i}{x_{i+1}^{n+1} - x_{i-1}^{n+1}} \cdot u_i^n$$

   where $c_i = (x_{i+1} - x_{i-1})u_i$ at $t = 1$

4. Repeat for each time step.

Chapter 6

A Location Based method

6.1 Introduction

Location-based methods are based on a one-to-one co-ordinate transformation between the computational space and the physical space, denoted by Ω_c and Ω respectively. To be more specific, location-based methods, are directly connected with time-dependent mapping.

One of the main methods in this category is the MMPDE approach. Taking the equidistribution principle a second PDE is constructed which describes the relation between the equidistributed co-ordinate x and a fixed co-ordinate ξ. The MMPDE is an additional PDE that is needed to locate the mesh.

Another location-based method is the *optimal transport method* which is analysed in [13]. The optimal transport method describes the natural generalisation of one-dimensional equidistribution. Given that the condition of equidistribution principle is satisfied, then the aim of the optimal transport method is to obtain an ideal mesh which can be derived by maintaining this mesh, to be as close as possible to a uniform mesh in a suitable norm. This method has been studied by Budd and his collaborators in [20]. In their work for illustrating the transformation from the computational to the physical domain the Monge - Ampere equation is used.

6.2 Moving Mesh Partial Differential Equations(MMPDEs)

MMPDEs are formulated using the *'equidistribution principle'* with an appropriate choice of monitor function. For simplicity MMPDEs are usually given in continuous form, allowing the comparison and the analysis studies based on the moving mesh method to be easily developed.

In the mathematical literature there are seven different types of MMPDEs. The most common MMPDEs in use are: *MMPDE1, MMPDE4, MMPDE5* and the *MMPDE6*.

MMPDEs cover a vast area of the location-based methods where actually they can be very beneficial for the numerical analysis of one-dimensional problems.

6.3 The Equidistribution principle and formulation of the MMPDEs in 1D

Huang, Ren and Russell in [3] carried out an extensive study on the effectiveness of the equidistribution principle, and how this approach can generate good numerical approximations.

Since the MMPDEs are directly associated with the equidistribution principle, for the purposes of the analysis we will go a step backwards and examine again how we derived the equidistribution principle and how corresponding to this assumption the MMPDEs have been developed.

Taking therefore again the continuous equidistribution principle from White in [2], given as

$$\int_0^{x(\xi,t)} M(x(\xi,t),t)dx = \xi \int_0^1 M(x(\xi,t),t)dx$$

and differentiating it with respect to ξ,

$$M(x(\xi,t),t)\frac{\partial}{\partial\xi}x(\xi,t)=\theta(t) \tag{6.1}$$

where $\theta(t)=\int_0^1 M(x(\xi,t),t)dx$. However, if we differentiate twice the eqidistribution principle with respect to ξ then this gives

$$\frac{\partial}{\partial\xi}\left(M(x,t),t)\frac{\partial}{\partial\xi}x(\xi,t)\right)=0. \tag{6.2}$$

Ren and Russell discussed in [8] the conservative form of a moving mesh equation. Differentiating (6.1), with respect to time using the chain rule leads to

$$\frac{\partial}{\partial\xi}(M\dot{x})+\left.\frac{\partial M}{\partial t}\right|_{\xi\text{fixed}} \qquad \frac{\partial x}{\partial\xi}=\dot{\theta}. \tag{6.3}$$

Furthermore, (6.3) can be transformed into a form with physical co-ordinates given as

$$\frac{\partial}{\partial x}(M\dot{x})+\left.\frac{\partial M}{\partial t}\right|_{\xi\text{fixed}}=\frac{M\dot{\theta}}{\theta}. \tag{6.4}$$

As a result, by comparison with the fluid dynamics literature, Ren and Russell show in [8] a physical interpretation of these moving mesh equations. On the other hand, in further studies by Huang, Ren and Russell [3], state without any supporting argument that the quantity $\theta(t)$ that holds in (6.4) is said to be inappropriate for generating numerical approximations. Consequently, they continue their studies by eliminating that term and create a series of Moving Mesh Partial Differential Equations (MMPDEs). They began their investigations by differentiating (6.2) with respect to time, giving

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

This equation after some rearrangements gives

$$\frac{\partial}{\partial \xi}\left(M\frac{\partial \dot{x}}{\partial \xi}\right)+\frac{\partial}{\partial \xi}\left(\frac{\partial M}{\partial \xi}\dot{x}\right)=-\frac{\partial}{\partial \xi}\left(\frac{\partial M}{\partial t}\frac{\partial x}{\partial \xi}\right) \tag{6.5}$$

which eventually leads to the development of the first MMPDE. According to [3], the MMPDE1 has a zero speed solution when $\frac{\partial M}{\partial t}=0$ and additionally the monitor function seems to be time-independent, implying that the mesh remains constant without moving. Moving along the same line, $\frac{\partial M}{\partial t}=0$ can be said to be a source of the mesh movement for this MMPDE. According to Huang et al this source quantity, is actually hard to calculate when used in actual cases.

In later studies of Huang et al, developed further MMPDEs. It is important to take into account that the equidistribution principle was the main criteria that the MMPDEs had to satisfy.

The moving mesh methods have been a significant tool for many authors in creating adaptive meshes for various applications. The most common in use MMPDEs in mathematical literature is the MMPDE1, MMPDE4, MMPDE5 and MMPDE6.

Chapter 7

Numerical Results

7.1 Numerical Results for the Velocity based method on a equidistributed mesh

In this chapter we present the numerical results that have been obtained, after applying the velocity-based moving mesh method to the one-dimensional PME (4.22). Including the self-similar initial conditions in the program allows us to investigate whether the numerical solution converges to the exact solution at a given time as the number of nodes increases. Unfortunately due to time limitations we were unable to program the location-based method.

To obtain adequate numerical results and sufficient representation of the solution, some further investigations are required before generating the pseudo code. Another issue that needs to be taken into consideration in advance is the size of timestep to be used, in order to avoid any tangling issues that might occur. Taking an initial equidistributed mesh we applied the velocity as given in (5.5) and compared the computed numerical solutions at different times.

7.2 Numerical Results for the Mass Monitor function

In this section we illustrate the approximated solutions using the mass monitor function

$$M(u) = u.$$

Firstly, by applying the Baines algorithm to the initial data (4.20) as given in [4], the mass is equidistributed in each computational cell. Then we aim to produce a good approximation to the self-similar solution

$$u(x,t) = \frac{1}{t^{\frac{1}{4}}}\left(1 - \frac{1}{4}\left(\frac{x}{t^{\frac{1}{4}}}\right)^2\right)^{\frac{1}{2}}$$

of the PME with $n = 2$.

We started the investigation using 20 computational nodes and applied a spatial size of 0.1, additionally setting the time-step to 0.1. However, the figures that we obtained led to the conclusion that the nodes were overtaking. The figures given below describe the results obtained after a 0.1 time-step has been applied to the problem. The numerical solution suffers from instability issues that result from the time scale which has been used.

Figure (7.1) depicts the numerical solution at $t = 1.1$.

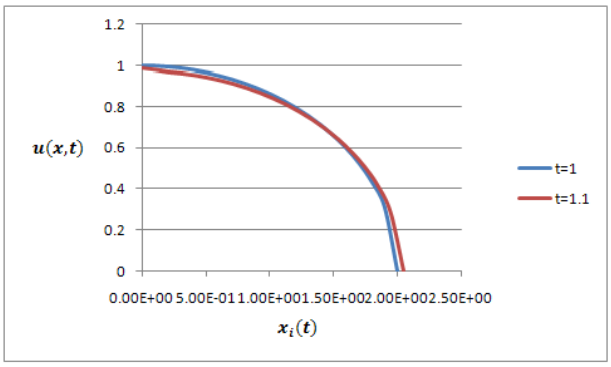


Figure 7.1: The numerical solution at t=1.0 and the corresponding at t=1.1 using the moving mesh method at $\Delta t = 0.1$

However, after a very small time-step further; at $t = 1.2$ it is clear from Figure (7.2) that the numerical solution begins to display some oscillations.

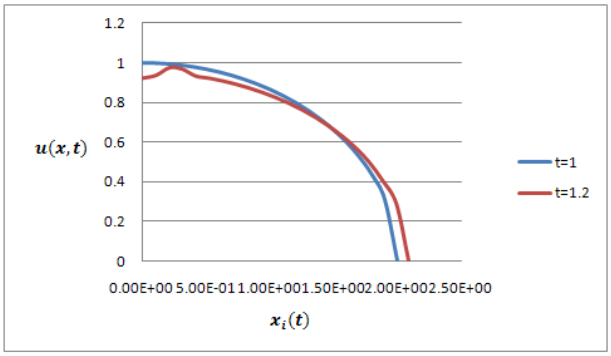


Figure 7.2: The numerical solution at t=1 comparing with the numerical solution at t=1.2

which leads eventually to an unstable solution. To make more obvious that the numerical solution blows up, the computational results of x_i with the corresponding u_s; that we obtained at a time $t = 2.0$, are given below

$u(x,t)$	$x(i)$
0.00E+00	8.09E-05
1235.415	0.434584
0.459442	-1.61E-04
1.964031	6.32E-04
313.1065	6.90E-04
285.8714	-6.20E-04
0.736031	-6.28E-04
-17.6051	0.184174
1.752725	9.84E-03
1.007389	2.22E-03
82.16696	2.68E-03
65.56568	-1.76E-03
-12.4293	-2.50E-03
1.577952	1.07E-02
1.799207	-7.73E-03
-16.8784	-3.74E-02
-1.73216	6.28E-03
2.157009	2.73E-02
2.109555	0.827727
2.261057	0.221383
2.374824	0.00E+00

Figure 7.3: Unstable results given at t=3.0

As you can observe from this table, the time-step of 0.1 led to large values of u_s implying that the numerical scheme is unstable. Therefore, before moving on to investigate further experiments for the efficiency of the moving mesh method, an advance investigation for the stability conditions and the timestepping to be used is required. This is a prerequisite before running the program again in approximating the accuracy of the moving mesh method.

In the next section, we will include a brief analysis about the stability that the moving mesh scheme requires in order to provide stable numerical solutions when applied to non-linear diffusion equation.

7.3 Timestepping

In order to find the new position of the nodes we need to apply a timestepping algorithm. For convenience we use an explicit method, which is easier to

implement. In fact for the purposes of this study we always applied the explicit forward Euler discretisation method

$$\dot{x^n} = \frac{x_i^{n+1} - x_i^n}{\Delta t} \tag{7.1}$$

$1 \leq i \leq N$ where Δt is the time-step that has been taken.On the other hand, this might lead to some other inefficiency in the numerical approximations solutions. Using the explicit method we have to take in consideration the stability issues as well. We need to be very careful with the choice of a selected time-step that we select to use for the numerical scheme.

Apart from the fact that the explicit forward Euler scheme is first order accurate, it is also only conditionally stable. This implies that a small time step is required to prevent instability and also to avoid any 'tangling problems'. Tangling issues occur when two nodes overtake each other. Since the area of each section interval is constant, this leads to a negative u which invalidates the method.

Most of the theory for the stability of the non-linear ODE system requires the study of Lipshitz constants. However in this case the analysis seems to be quite complicated. More directly, we consider a simpler technique to avoid tangling which still gives some justification for the chosen value.

Given that $v_{i+1} - v_i$ represents the *relative velocity* and then $x_{i+1} - x_i$ the *relative position* respectively, then in order to avoid any tangling, the sufficient condition for non-tangling

$$\Delta t < \left| \frac{x_{i+1} - x_i}{v_{i+1} - v_i} \right|, \qquad \forall_i$$

must be satisfied (which changes at each time step).

Therefore, regarding the first case above when we applied a time step of 10^{-1} was apparently not sufficient to satisfy this inequality, leading as a result to an unstable scheme. Therefore we ran the program again using a smaller time stepsize based on trial and error.

7.3.1 Numerical Results for a smaller time-step

In this section we run the program that we used above but this time we use a time step of $\Delta t = 10^{-3}$ in order to obtain a more reliable set of results for the numerical solution. We applied the same timestepping algorithm as before, using 20 computational nodes and as before a final time step of 3.0. In contrast with the previous experiment, in this case we needed 2000 time steps. Any bigger value for the timestep will produce an unstable representation of the numerical approximation.

The figures below show the solutions obtained for the PME (4.22) when $n = 2$. Figure (7.4) represents the behaviour of the grid points and how they move with time over the whole domain.

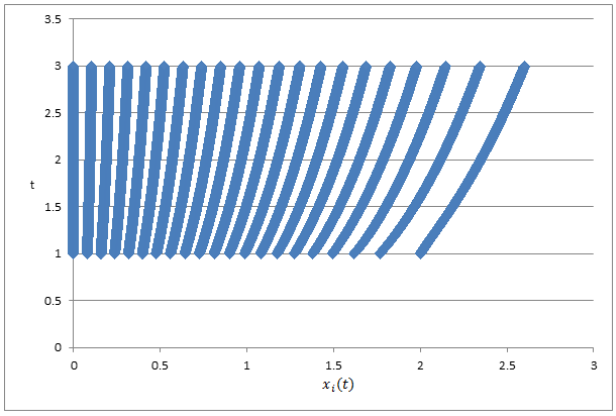


Figure 7.4: The mesh trajectories

The Figure (7.5) is exactly the same as before since it is given by the initial condition when t=1.0. Figures (7.6) and (7.7) show that the approximation solutions become smoother and the mesh movement has been succesfully carried out.

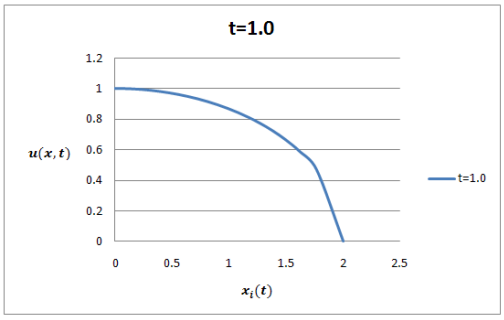


Figure 7.5: The initial solution at t=1.0

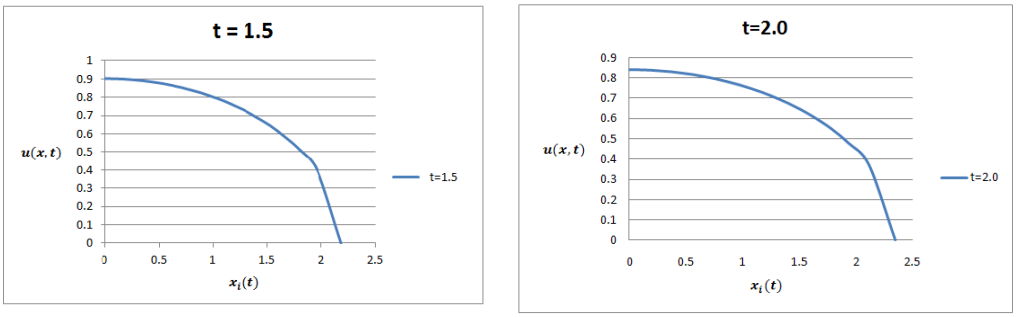


Figure 7.6: Left: Numerical solution at t=1.5, Right: Numerical solution at t=2.0

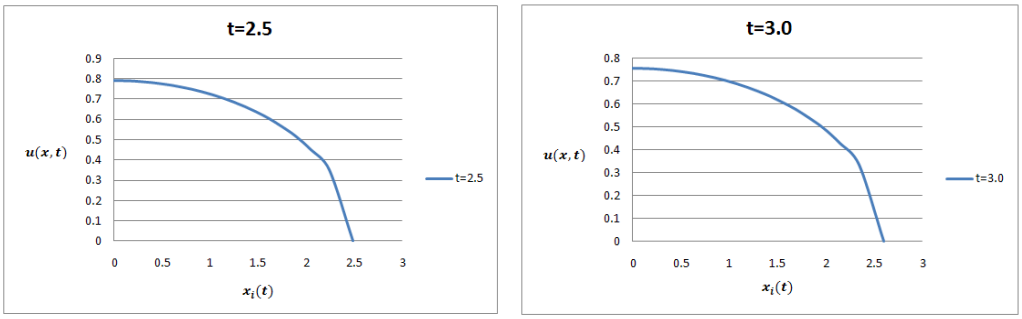


Figure 7.7: Left: Numerical solution at t=2.5, Right: Numerical solution at t=3.0

Figure (7.8) gives an overview of how the mesh moves after applying the moving mesh method. It is obvious that the nodes remain evenly spread

outwards and the numerical solutions become more flat with time. It is also important to notice that no node overtaking occurs.

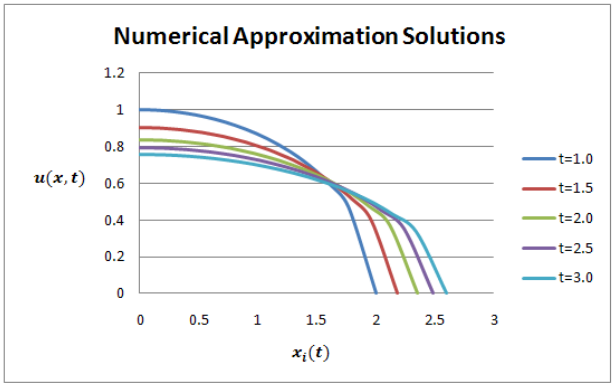


Figure 7.8: The numerical approximation solutions at final time t=2.0, using the moving mesh method

According to these figures we can clearly observe that the instabilities and the undesirable features in the numerical solutions have been removed when a smaller size of time-step has been taken.

Figures (7.9) and (7.10) depict the numerical solutions against the exact solutions, constructed at 50 computational nodes using a time-step of $\Delta t = 10^{-3}$. The numerical solutions almost overlap the exact solutions while moving with time, indicating that using the moving mesh method provides quite accurate numerical approximations.

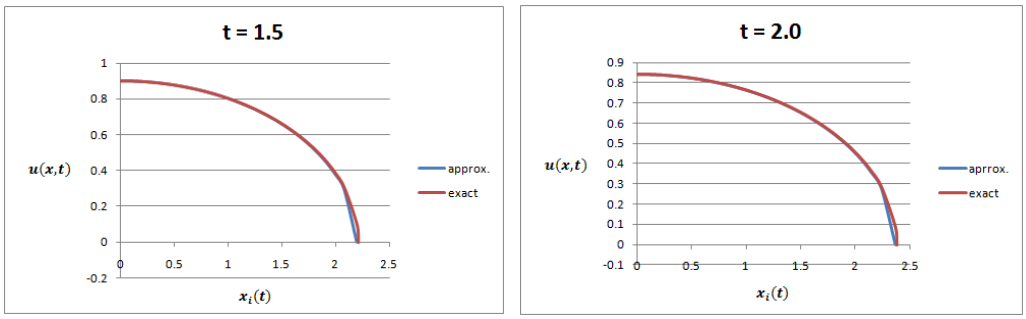


Figure 7.9: Left: Numerical solution against analytical solution at t=1.5, Right: Left: Numerical solution against analytical solution at t=2.0

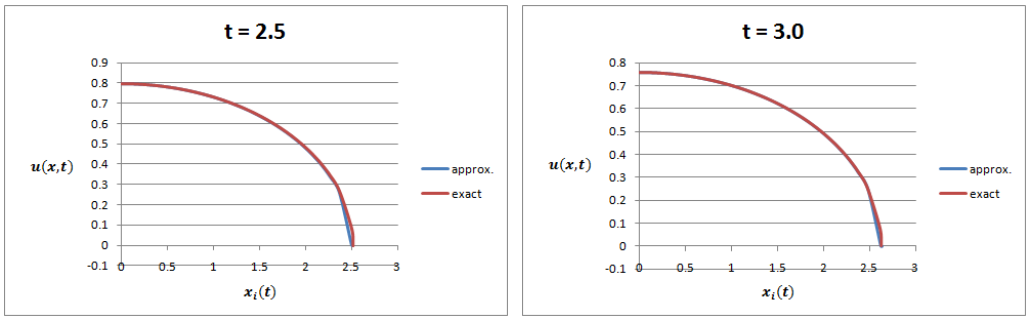


Figure 7.10: Left: Numerical solution against analytical solution at t=2.5, Right: Numerical solution against analytical solution at t=3.0

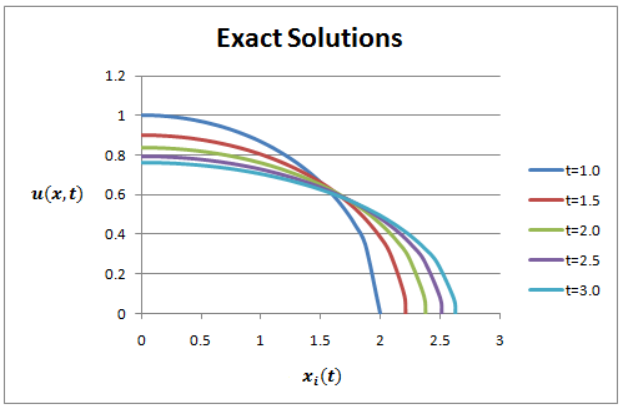


Figure 7.11: The exact solutions throughout the life of the graph, using the moving mesh method

The technique of self-similar solutions both enhanced our investigations but also enabled us to calculate the accuracy of the moving mesh method. In the next section two techniques are introduced in order to compute the error occurred between the numerical solution and the exact solution.

7.4 Error Calculations

In order to study whether the scheme that we applied gives an efficient and useful representation of our solution, we need to examine the convergence of the numerical moving mesh method.

Numerical approximations do not give exact solutions, but they can be used in order to obtain a solution that is close to the exact solution while maintaining a reasonable range of errors. We need to check the accuracy of the velocity-based moving mesh methods. In order to calculate the accuracy between the numerical approximation solution and the exact solution we need to apply an 'error' formula that calculates the difference within those solutions.

In mathematical theory there are several error calculation techniques available to check the accuracy of the scheme used. For the purposes of this dissertation we used the *'absolute $l_2 norm$'* and the *'relative error'* in order to examine the accuracy.

Applying the *'absolute error'* measure

$$l_2 = \sqrt{\sum_i (u_i - u_{exact})^2}$$

to the program that calculates the numerical approximation, we investigate the error that occurs at every timestep. The u_i are the values obtained from the numerical adaptive method while u_{exact} is defined by the corresponding values of the exact solution. Increasing the number of points while decreasing the size of the timesteps will generate a more accurate solution.

The Figure (7.12) shows the absolute error between the numerical approximation solution and the exact solution.

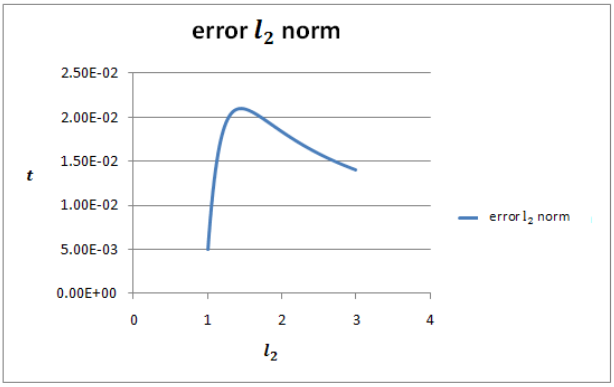


Figure 7.12: The behaviour of the absolute error calculation

The absolute error starts by increasing but then as time progresses it decreases since the numerical solution become more flat in time.

The calculation for the *'relative error'* is generally expressed as a percentage and it works out the percentage ratio between the absolute error and the true value solution, the u_{exact}. The relative error is determined using the following formula

$$\text{relative error} = \sqrt{\frac{\sum_i (u_i - u_{exact})^2}{\sum_i (u_{exact})^2}}$$

Figure (7.13) shows the relative error between the numerical approximation solution and the exact solution. This error increases slowly with time.

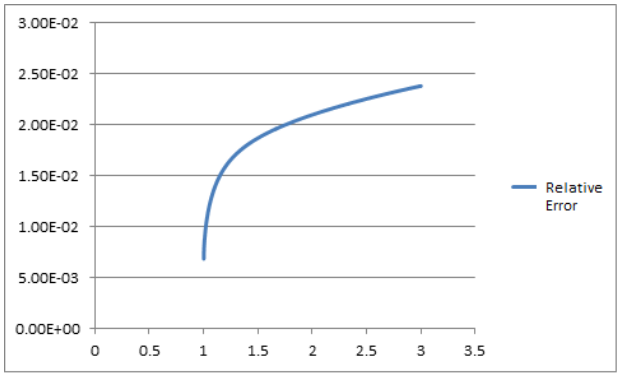


Figure 7.13: The behaviour of the relative error calculation

The relative error in our case appears to be a more reliable 'error measure' to take, since the numerical solutions become more flat with time. In order to verify the accuracy of the relative error we performed a small test. We run the programme using a timestep of $\Delta t = 10^{-5}$ using different number of computational nodes. We set a final time of t=1.02.

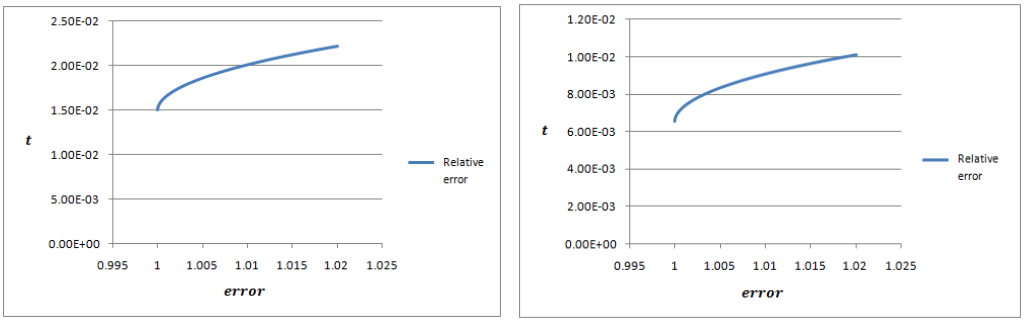


Figure 7.14: Left: Relative error calculation using $N_x = 10$, Right: Relative error calculation using $N_x = 20$

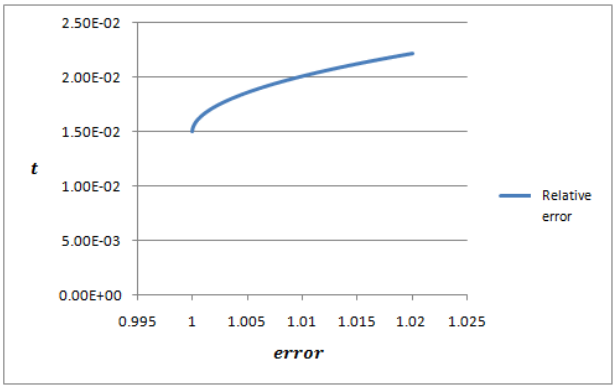


Figure 7.15: Relative error calculation using $N_x = 40$

The results obtained at the final time are given in the Figure (7.16) below

Relative Error

t	Δt	Nx	Relative error	Rate of convergence
1.02	10^{-5}	10	2.22^{-02}	
1.02	10^{-5}	20	1.01^{-02}	***2.198***
1.02	10^{-5}	40	4.51^{-03}	***2.240***

Figure 7.16: The behaviour of the relative error calculation at different number of computational nodes

From Figure (7.16) we can see that when the number of computational nodes increases, then the relative error converges with a rate of convergence

greater than one.

The approximate solution seems to be accurate and stable implying that the velocity-based adaptive method that we used gives good results to self-similar solutions to the problem under consideration. In general, the PME possesses solutions for which an interface can remain fixed for a finite time and then start moving. The time for which the interface remains stationary is called the waiting time and solutions which exhibit such behaviour are called waiting time solutions. However, when we use a self-similar solution waiting times do not occur

Chapter 8

A Velocity-based method using an arc-length monitor function

In this section we study the efficiency and accuracy when the arc-length monitor function is used to generate numerical solutions of the PME. The computational solutions have been calculated using the same algorithm and theory as applied in the previous section. However, for this approach the velocity needs to be recalculated using the different monitor function, which introduces as a result an unknown variable $\theta(t)$ for the total arc-length within the calculations. We first go over the theory and consider the changes that are required.

8.1 Evaluation of the mesh velocity

The arc-length monitor function is defined as

$$M = \sqrt{1 + u_x^2} \tag{8.1}$$

and will be applied to the PME (4.22) in the interval $(0, 2)$ with the initial condition

$$u = \left(1 - \frac{x^2}{4}\right)^{\frac{1}{2}} \tag{8.2}$$

and boundary conditions $u = 0$ at $x = 2$ and $u_x = 0$ at the origin, as before.

The total arc-length is defined as

$$\theta(t) = \int_0^2 \sqrt{(1 + u_x^2)dx} \tag{8.3}$$

which is not constant in time, so it is inconsistent to make the partial arc-lengths constant in time. However we can make the arc-length *fractions*

$$\frac{1}{\theta(t)} \int_0^x \sqrt{(1 + u_x^2)}dx = \text{constant in time, say} c_i \tag{8.4}$$

at the expense of the additional variable $\theta(t)$.

Note that

$$c_i = \frac{\int_0^{x_i} \sqrt{(1 + u_x^2)}dx}{\int_0^{b(t)} \sqrt{(1 + u_x^2)}dx} = \frac{\text{partial arc length}}{\text{total arc length}} \tag{8.5}$$

indicating that the constants; c_i, represent a fraction of the arc length used.

Using (8.4) we can compute the constants, c_i, at time t=1.0 at every cell of the mesh.

In general the derivation of the the mesh velocity is consistent with the structure that we explained when a mass monitor function was applied, apart from the fact that in this case we have the extra parameter $\theta(t)$.

Firstly we rearrange equation (8.4) to

$$c_i\theta(t) = \int_0^x \sqrt{(1 + u_x^2)}dx. \tag{8.6}$$

Then, differentiating (8.6) with respect to t will give us

$$c_i\dot{\theta}(t) = \frac{d}{dt} \int_0^x \sqrt{(1 + u_x^2)}dx. \tag{8.7}$$

We then apply *'Leibnitz Integral Rule'* to give

$$\begin{aligned} c\dot{\theta} &= \frac{d}{dt}\int_0^x \sqrt{(1+u_x^2)}dx \\ &= \int_0^x \frac{\partial}{\partial t}\sqrt{(1+u_x^2)}dx + \left(\sqrt{(1+u_x^2)}\cdot v\right)\Big|_0^x \\ &= \int \frac{\partial}{\partial t}\sqrt{(1+u_x^2)}dx + \sqrt{(1+u_x^2)}v \end{aligned} \tag{8.8}$$

and then using the *'chain rule'* we finally derive

$$c\dot{\theta} = \int_0^x \frac{u_x}{\sqrt{(1+u_x^2)}}u_{xt}dx + \sqrt{(1+u_x^2)}\cdot v \tag{8.9}$$

since $v = 0$ at $x = 0$.

In addition we need to adapt (8.9) to the problem that we are dealing with, which in our case is the PME with zero boundary conditions. Taking that the PME is given as

$$u_t = (u^2 u_x)_x \tag{8.10}$$

then differentiating (8.10) with respect to x leads to

$$u_{xt} = (u^2 u_x)_{xx}. \tag{8.11}$$

If we thus substitute (8.11) back to the equation (8.9) then

$$c\dot{\theta} = \int_0^x \frac{u_x}{\sqrt{(1+u_x^2)}}(u^2 u_x)_{xx}dx + (\sqrt{(1+u_x^2)})v \tag{8.12}$$

Finally by rearranging the (8.12) we get

$$v(x) = \frac{1}{\sqrt{(1+u_x^2)}}\left[c\dot{\theta} - \int_0^x \frac{1}{\sqrt{(1+u_x^2)}} u_x (u^2 u_x)_{xx} dx\right] \tag{8.13}$$

so that we have a general form of the mesh velocity. Equation (8.13) illustrates the general velocity that has been applied on the interior computational nodes in order to provide the relocation of the grid points over time. However, we are not able to calculate the mesh velocity yet, since the parameter $\dot{\theta}$ is still unknown at this stage, but we are nearly there. The parameter $\dot{\theta}$ equals the rate of change of the arc length over each time-step. To calculate the $\dot{\theta}$ we need to refer back to one of the properties of the PME.

Taking into consideration that the PME is *mass conservative* then this enables us to state that the velocity at the last boundary point $x = 2$ can be defined as before, as

$$v_b = -uu_x = -\frac{1}{2}(u^2)_x \tag{8.14}$$

Matching the left hand side of (8.14) to the general velocity (8.13) at the boundary, leads to

$$-uu_x = \frac{1}{\sqrt{(1+u_x^2)}}\left[\dot{\theta} - \int_0^b \frac{1}{\sqrt{(1+u_x^2)}} u_x (u^2 u_x)_{xx} dx\right] \tag{8.15}$$

By rearranging (8.15) it will eventually give us the computational value of $\dot{\theta}$, which now can be substituted back in (8.13) and finally allow us to calculate the mesh velocity.

In the numerical method the velocity v_i at a node is computed from (8.13) at $x = x_i$. The new meshpoints are then found from the explicit Euler method. The initial grid is obtained by equidistribution of the arc-length.

8.1.1 How to calculate the solution

To calculate u_i we applied the constant arc-length formula which is coupled with *Pythagora's theorem* given as

$$\Delta x^2 + \Delta u^2 \quad = \quad (\theta c_i)^2. \tag{8.16}$$

Since we seek the computational $u's$, we rearrange (8.16) and make the Δu the subject. This leads to,

$$\Delta u = \sqrt{(\theta c_i)^2 - \Delta x^2} \tag{8.17}$$

and (8.17) can be used in the algorithm in order to find the computational positions of the grid points.

Referring back to the boundary conditions, we already know that $u = 0$ at $x_N = 2$. This implies that for calculating the computational $u's$ we must begin at the last node, at x_N, and then go backwards by calculating the Δu and hence the unknown $u's$ at the previous nodes.

The Figure (8.1) shows how using the Pythagoras'theorem we can calculate the $u's$ position.

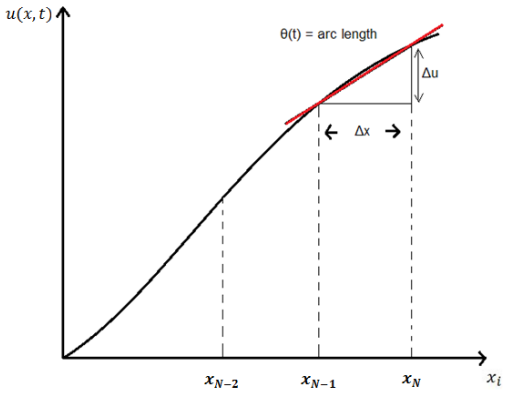


Figure 8.1: The calculation of the position of the $u's$

The full algorithm to obtain the PME numerical solution using an arc-length is given below.

Algorithm:

Given a mesh with meshpoints x_i and values u_i^n ; $i = 0, \ldots, N$, at $t = 1$

1. Using (8.3) calculate the total arc-length, $\theta(t)$ by evaluting

$$\theta(t) = \sum_{i=0}^{N-1} \left(1 + \left(\frac{u_{i+1} - u_i}{x_{i+1}^n - x_i^n}\right)^2\right)^{\frac{1}{2}} \cdot \Delta x$$

   where Δx is the initial spatial step and compute the constants, c_i, from (8.5).

2. Compute the mesh velocity at the last boundary x_N using (5.10). This will determine the unknown value of $\dot{\theta}$. When $\dot{\theta}$ is obtained, then we use (8.13) in order to calculate the mesh velocity at interior nodes.

3. Using the Forward Explicit Euler scheme, compute the updated mesh x_i^{n+1}.

4. Compute the updated solution of u_i^{n+1} from (8.17) by applying

$$u_i^{n+1} = u_{i+1}^{n+1} + ((\theta \cdot c_i)^2 - (\Delta t)^2)^{\frac{1}{2}}$$

   and $u_N^{n+1} = 0$.

5. Repeat for each time step.

8.1.2 Numerical Results for arc-length monitor function

In this section we run a FORTRAN program using the algorithm given above. We used a really small time step of $\Delta t = 0.0005$. The programme was constructed using 40 computational nodes and a final time of $t = 0.5$.

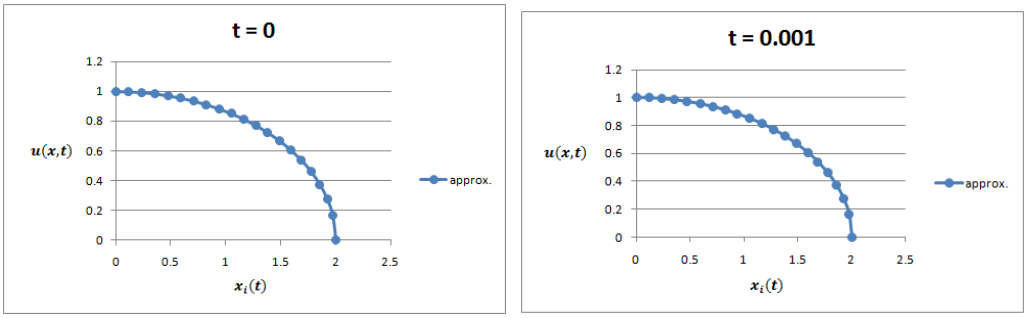


Figure 8.2: Left: Numerical solution using an arc-length monitor function at $t = 0$, Right: Left: Numerical solution using an arc-length monitor function at $t = 0.001$

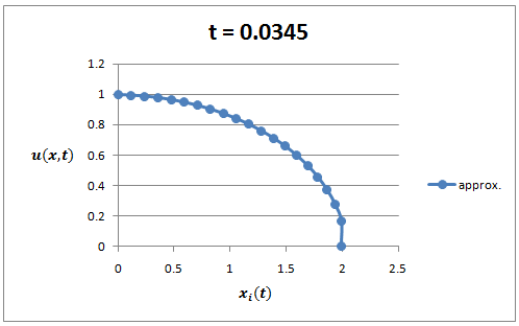


Figure 8.3: Numerical solution using an arc-length monitor function at $t = 0.0345$

These figures show that initially the solution seems to be moving fine but only for a while. As time progresses the solution seems to overturn, implying that the boundary velocity lags the other nodal velocity. A possible explanation for this problem is the fact that the function that we used is infinately steep at the boundary. Also, we noticed that the arc-length calculated at the last cell is longer than the previous cells, implying that equidistribution principle for the initial grid might not be the most appropriate technique to use in this case. We applied a central finite difference method to compute the numerical solutions. However, this caused some problems in calculating the last node position at the boundary. Due to limitations on the domain we were forced to use an one-side finite difference method although, this was likely to cause some inefficiencies to the numerical solutions that we obtained.

Chapter 9

Summary and Further Work

9.1 Conclusions

Grid adaption and moving meshes have experienced considerable development in recent years and are becoming a useful technique in the numerical solution of a wide variety of applications. Adaptive methods have a critical role in the area of numerical analysis for distributions involving steep or sharp solution variations. These techniques effectively contribute in the improvement of the numerical solutions of PDEs.

This final chapter summarises the work carried out and presents some of the output obtained. Additionally, discussion regarding the efficiency of the results and findings is made. Finally some possible suggestions for future avenues of research will be suggested.

Initially we started our study by giving an introduction of the equidistribution principle which is the main source of the moving mesh method. We showed how the equidistribution principle gives different meshes for a given function using different choices of monitor function. In this dissertation we concentrated mainly on the velocity based method using a mass monitor function and an arc-length monitor function.

A special case of similar solutions has been applied for the purposes of this study, known as 'self-similar' solutions. This is another advantage of the properties of the PME. A self-similar solution has been used as an initial

condition and implemented in both equidistribution and the moving mesh algorithm, aiming to present the effectiveness of this scheme.

In Chapter Seven, using a mass monitor function we applied the moving mesh method comparing with a self-similar solution of the PME. The moving mesh method needs a very small time step to be taken, due to stability reasons and then seems to be quite accurate. The numerical solutions evolved nicely and become flat with time. Using an error measure, a relative error calculation, indicates that the moving mesh method is an accurate numerical approximation since the error converges with a rate of convergence greater than one.

In Chapter Eight we continued our investigations on the moving mesh, using an arc-length monitor function. In this experiment we found some limitations in calculating the velocity on the last boundary point. Even though at the beginning the numerical solutions seemed to be moving normally for a while, as time progresses the numerical solutions begin to overturn. Another constraint that we had using this technique was the equidistribution of the arc-length of the function under consideration. The function we applied was very steep at the boundary and for that reason the arc-length of the last cell was larger.

9.2 Further Work

In this section we consider some further aspects that have not been fully analysed in this dissertation and could be taken into account in further research for more accurate results.

9.2.1 Timestepping

For the purposes of this dissertation we used the Forward-Euler method in order to apply the time-step in the scheme. The time-step is required due to stability condition.

A possible suggestion for further investigations could be to examine the implementation of a semi-implicit time-stepping scheme which allows a larger

size of time step to be taken without causing any tangling problems. However, the accuracy of using a semi-implicit approach needs to be tested. Additionally, other time-stepping schemes might be taken into account that are more accurate than the Forward-Euler method, such as the Runga-Kutta method.

9.2.2 Initial grid distribution

To test the moving mesh method we used an equdistributed initial grid. Some further work that could be considered is to start our investigations using another initial mesh in order to check if the solution obtained will be affected, for example when applying the moving mesh method with an arc-length monitor function. The moving mesh method does not actually depend on the initial mesh being equidistributed.

9.2.3 Initial function

Using another initial function apart from the self-similar solution, will cause waiting times which can be investigated.

9.2.4 MMPDEs

Unfortunately, due to limitation of time we did not manage to present some numerical solutions using the location-based method. Therefore, a future study could be to investigate the applications of the MMPDEs. This will enable us to make a more critical comparison with the velocity-based methods.

9.2.5 Two-dimension

In this dissertation we applied the moving mesh method on one-dimensional problems. Further investigation that could be taken into consideration is to apply the moving mesh method to problems in 2D. Consequently, it will allow us to obtain a broader idea of the efficiency of the moving mesh method.

Bibliography

[1] E.A.Dorfi and Drury (1987) *Adaptivity with moving grids*-Chris J.Budd.

[2] A.B.White, On the selection of an equidistributing mesh for two point boundary-value problems. *SIAM Journal on Numerical Analysis. Vol.16, No.3, 472-502*, 1979

[3] W.Huang,Y.Ren and R.D.Russell. Moving mesh partial differential equations (MMPDEs)based on the equidistribution principle.*SIAM Journal of Numerical Analysis. Vol.31, No.3,709-730*, 1994

[4] M.J.Baines. Grid adaption via node movement. Applied Numerical Mathematics 26,77-96, 1998

[5] C.De Boor. Good approximation by Splines with Variable knots II, *Springer Lecture Notes Series 363*, 1973

[6] C.J.Budd, W.Huang, and R.D.Russel. Adaptivity with moving grids *Acta Numerica, 1-131,* 2009

[7] W.Cao, W.Huang, and R.D.Russell. A study of monitor functions for two-dimensiolnal adaptive mesh generation. *SIAM J.Sci.Comput., 20(6): 1978-1994,*1999

[8] Y.Ren, and R.D.Russell. Moving mesh techniques based upon equidistribution, and their stability. *SIAM Journal of Scientific Statistical Computing. Vol.32, Issue 3,*1992

[9] D.A.Anderson. Adaptive mesh schemes based on grid speeds *AIAA PAPER 83-1931, 311-318*, 1983

[10] S.Adjerid and J.E.Flaherty. A moving finite method with error estimation and refinement for one-dimensional time dependent partial differential equations. *SIAM Journal on Numerical Analysis. Vol.23, Issue 4, 778-796*, 1986

[11] J.M.Coyle, J.E.Flaherty, and R.Ludwig. On the stability of mesh equidistribution strageties for time-dependent partial differential equation. *Journal of Computational Physics, 62, 25-39*, 1986

[12] S.Li, L.Petzold, and Y.Ren. Stability of moving mesh systems of partial differential equation. *SIAM Journal of Scientific Computing. Vol.20, No.2, 719-738*, 1999

[13] Budd, C.J. and William, J.F Moving mesh generation using the Parapolic Monge-Ampere equation. *SIAM J.Sci.Comput.*, 2008

[14] D.G.Aronson. The porous medium equaion. In *"Nonlinear Diffusion problems"*, Lecture notes in mathematics, No 1224, Springer-Verlag, 1986.

[15] A.S.Kalashnikov. Some problems of the qualitative theory of non-linear degenerate second order parabolic equations. *Russion Mathematical Surveys. 42, 2, 169-222,* 1987

[16] J.L.Vazquez. An inroduction to the mathematical theory of the porous medium equation. *Shape Optimisation and free Boundaries. Kluwer Academic, 347-389,* 1992

[17] K.Vafai and C.L.Tien. Boundary and Inertia Effects on flow and Heat transfer in porous media. *Int.J.Heat Mass Transfer. Vol.24, 195-203,* 1981

[18] Vazquez, J.L The porous medium equation: Mathematical theory *Oxford University Press*, 2007

[19] Tamsin E.Lee Modelling time-dependent partial differential equations using a moving mesh approach based on conservation, *The University of Reading*, 2011

[20] B.V.Wells. A moving mesh Finite Element Method for the Numerical Solution of Partial Differential Equations and Systems. *PhD thesis, University of Reading*, 2004

[21] M.J.Baines, M.E.Hubbard and P.K.Jimack. Velocity-based moving mesh methods for Nonlinear Partial Differential Equations, Commun. Comput. Phys, 10, 509-576, 2011