UNIVERSITY OF READING
Department of Mathematics

University of Reading

MSc Dissertation

Numerical Approximation of a Quenching Problem

Author:
Michael Conland

Supervisor:
Professor Michael J. Baines

August 21, 2010

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

Signed:

Date:

Abstract

This dissertation contains a study of the effectiveness of using an r-adaptive moving mesh method on the quenching equations $u_t = \frac{1}{a^2}u_{xx} + f(u)$ and $u_t = \frac{1}{a^2}\nabla^2 u + f(u)$, where $f(u) = \frac{1}{(1-u)^\theta}$, which under the right conditions are known to blow up in finite time in the centre of the domain on which they are being solved. Preliminary studies are carried out using fixed mesh methods, before using u_t as a monitor function for calculating nodal velocities in both a one and a two dimensional case. Numerical results are compared and conclusions drawn on the use of u_t as a monitor function.

Acknowledgements

The greatest of thanks must be directed towards my supervisor Mike Baines, whose door was never closed. Without his continued support, wealth of knowledge, ideas and that draw which never seems to run out of scrap paper, none of this would have been possible.

Contents

Chapter 1

An Introduction

1.1 What is Quenching?

Quenching can refer to two very different phenomena. The first of these refers to physical processes, often relating to the cooling of materials. For example, steel and other alloys are often quenched by heating them to a certain temperature and then rapidly cooling the metal to harden it. However, this is not the form of quenching being considered in this dissertation, which is a mathematical idealisation of quenching and therefore it will be with reference to partial differential equations (PDE) that we discuss quenching.

It is often seen when integrating a partial differential equation that the solution will change rapidly and result in the solution tending to infinity. Often this is caused by the rapid change in values of one particular term within the partial differential equation. For example, the equation

$$u_t = u_{xx} + f(u), \tag{1.1}$$

posed by Kawarada [8] (where $f(u) = \frac{1}{1-u}$) and subsequently Liang et al [9] is known to quench due to the rapid growth of the u_t term.

Quenching can also arise in different forms. In the case of references [8] and [9], both show that the problem being considered is symmetrical and that the quenching takes place at a single point in the centre of the one dimensional domain. Karawada [8] also found conditions on which the above problem will quench within finite time. Deng and Levine [7] summarised Karawada's work, stating that if equation (1.1) is considered in the domain $0 <= x <= L$, then if $L > 2\sqrt{2}$, $u(L/2, t)$ will reach unity in finite time and if that is the case, $u_t(L/2, t)$ will become unbounded in finite time. Put more crudely, equation (1.1) will generate quenching/blow up in the centre of the domain, in finite time, assuming the domain's length is greater than $2\sqrt{2}$.

However, it is not always the case that quenching occurs at a single point in a domain. Chan and Ke [5] and Chan [4] found that, for equation (1.1), if $f(0) > 0$, $f' \geq 0$ and $f'' \geq 0$, the solution will eventually reach the quenching point throughout the domain. This is referred to as complete quenching. Studies have also been undertaken as to what occurs post quenching, as (according to [9]) the post quenching results can also have a physical meaning.

On another note, despite the earlier statement that the quenching referred to in this paper will be mathematical, mathematical quenching is often related to the modelling of physical processes. Budd et al [2] state that quenching, amongst other mathematical phenomena, is known to appear in problems concerning gas modelling (Liang et al base their paper on gas within a porous wall), combustion, detonation and mathematical biology amongst others.

This project will centre around the solving of a quenching problem using an adaptive method and thus we must consider why the use of an adaptive method is particularly relevant here, since it is possible to find a quenching solution using a fixed mesh. Obviously a numerical method is required because there is no analytic solution to equation (1.1) and despite it being possible that a complete quenching solution exists, this only occurs under specific conditions. Often, the quenching will occur at a single point and the issue with a fixed mesh method is that, without high resolution, the quenching point can be missed entirely. As such, unless the quenching point is known before the mesh is created, a very large number of nodes is required to guarantee the uncovering of a quenching point.

This is where the adaptive methods become ideal, especially the h and r-methods (which will be explained shortly). By starting with a fixed grid and either adding extra nodes in or shifting them based on the solution, one can create a high resolution of nodes around the point where the greatest rate of change (and hence the quenching point) occurs. Therefore the advantage of using an adaptive method for a quenching problem is that the mesh resolution about the quenching point is increased, thus increasing the chance of actually locating the quenching point (assuming we do not know its location). Moving mesh methods also have the possibility of being more efficient than their mesh refinement counterparts, as in theory they should use fewer nodes and therefore less computational effort.

1.2 Adaptive Methods

Generally there are three forms of adaptive method, these being the mesh refinement (h), order enrichment (p) and mesh movement (r). This dissertation will centre around the use of a mesh movement method with a fixed number of nodes. Rather than using the more well known mesh refinement methods, which add extra points into the mesh where they are required, the number of nodes will be set at the start of the method and these will be shifted to the areas of the domain which require them.

There are both positives and negatives to the use of the mesh movement methods rather than the refinement methods. Budd et al [2] state that h-methods have been studied in great depth over many years, so much so that these methods now exist in commercial codes. r-methods on the other hand have seen far less interest and as such there are fewer studies to consider. Adding that adaptive methods can suffer greatly from a requirement for high levels of computation. Generally, the nodes in the mesh are often shifted using another PDE, as well as the PDE which is actually being solved upon the mesh. As such the computational cost can be rather large, since two different systems are being considered. This introduction of another PDE can also lead to a need to solve a stiff system, thus increasing the computational costs again. A further complication can arise when nodes move too quickly, forcing parts of the domain to overlap and thus invalidating the solution.

However, despite these negative aspects, there are still arguments for the use of r-adaptive methods. Although the computational cost of shifting the mesh can be high, a high computational cost is just as much an issue in h-methods as it is in r-methods. The h-method may not need to calculate a solution of a PDE to shift the nodes, but it does have to calculate how new nodes (or the removal of nodes) get incorporated into the solution. It is also stated in [2] that r-methods lend themselves well to problems in which the spatial and temporal length scale are very different, suggesting that this should result in an r-adaptive method being suitable in the solution of a quenching problem.

1.3 Monitor Functions and Methods For Solution

Having briefly explained both quenching and the type of method being considered here, the means of calculating the changes to the mesh should be considered. There are two aspects of the r-adaptive method which dramatically change the solution of the PDE. These two aspects are the monitor function and subsequently the class of the method.

If we consider two examples for this, we can see the scope for different methods for the solution. The first of these is the aforementioned paper by Liang et al [9]. This is technically an h-method, but the monitor function used serves as a good example for the different choices which can be made. In [9] a monitor function is used which is

$$\mu(u_t) = \sqrt{1 + u_{tt}^2},$$

an arc length monitor function. It should be noted that in [9], equation (1.1) is being solved by considering the domain to have some number of equally spaced nodes in the spatial direction. However, it is the time stepping in this paper which is adaptive and this is what the monitor function is used for. The monitor function governs the time step at each of the spatial nodes and alters it based on values of u_t.

There are many different options when it comes to using monitor functions in refinement methods. One which is particularly relevant here is that used by Baines et al [1], as well as the various others noted in [2]. This monitor function is the mass (or area) under the function u in the domain and the study in [1] is particularly relevant to this paper as the method used here is very similar. The method used in [1] of conserving the mass under the curve in time equates to preserving the area under the curve between two nodes to be its initial values. The nodes are moved by forcing each of these areas to retain their area (relative to the total mass under the curve) at each time step.

The subsequent part of such schemes is again summarised by Budd et al in [2]. Two options are referred to as velocity and location based methods. Velocity based methods calculate the mesh velocity to move the mesh, similar to Lagrangian methods. These methods have spanned a vast amount of literature alone and contain studies into how to avoid mesh tangling, solving using finite elements, methods based on conservation laws (GCL method, explained by Cao et al [3]) and the use of penalty functions to avoid singularities. However, there also exist studies which are centred on avoiding the use of penalty functions as well as those advocating the use of them.

Location-based (according to [2]) methods on the other hand, employ a monitor function to determine the moving of the coordinates and the movement of the mesh will be based on a minimisation.

1.4 Aims of This Study

The final part of this introduction is to briefly explain the aims of this study. The main aim is to complete the derivation and subsequently the coding of a velocity-based, r-adaptive moving mesh method. There will be two main components to this, the first of which will be to solve the problem posed by Kawarada and Liang et al (equation (1.1)) by means of using the scaled version from [9]. This will be explained later. The second aim is to complete a two dimensional, finite element version of this method, applied to a circular domain using a modified version of the scaled equation posed by [9].

Clearly it will be far from simple to calculate the truncation errors created by such schemes using analytic methods and as such, two preliminary tests will be carried out in section 2. The one dimensional test will use exactly the same PDE as that of the one dimensional adaptive case, whereas the two dimensional case will use a radial version of the PDE, rather than a genuinely two dimensional version. Since the problem is known to be symmetric (from [9]), it must be radially symmetric in two dimensions. As such, providing a sufficient number of nodes are used, the results of a radial and a genuinely two dimensional version of the same equation should be much the same. Both the one and two dimensional tests will use standard, explicit finite difference schemes to solve the PDEs.

The preliminary tests using fixed meshes will set the benchmark for the adaptive methods, the hope being that the adaptive methods show themselves to be more efficient or produce more effective results around the quenching point than that of their standard finite difference counterparts.

The two adaptive methods shall then be considered in sections 3 and 4 respectively. They will use the same initial conditions as the preliminary tests and in the case of the one dimensional problem, exactly the same equation. The two dimensional case will differ slightly from the preliminary tests, as a genuinely two dimensional equation will be considered. However, much of the solution will be considered radially and the output from this method will also be in radial form.

Finally, conclusions on the methods will be drawn in section 5 and then suggestions for further work, based on what has been found in this study, will be made.

Previous work has been carried out on a very similar problem to this, using the method from [1], by Cole [6]. The equation $u_t = u_{xx} + u^2$ was considered, with u as the monitor function. Her study proved that the method from [1] could be applied to such a PDE and thus, the hope is that this similar method will prove successful in solving the PDE being used in this dissertation.

Chapter 2

Preliminary Tests

The reasoning behind performing these basic tests is to give benchmark values using schemes with known error estimates. The equation proposed in [8] has no analytic solution and therefore there is no definite solution to test the adaptive methods against. As such, two preliminary tests will be undertaken, one for the one dimensional case and a second to give an approximation to a two dimensional solution, since the the problem posed is radially symmetric. Both tests will be carried out on uniform grids and with fixed time steps using standard finite difference methods.

2.1 A Test in One Dimension

The problem being solved here is exactly that of [9] p5, which is simply a scaled version of the equation posed by Kawarada in [8]. Thus the equation being solved is

$$u_t = \frac{u_{xx}}{a^2} + f(u), \tag{2.1}$$

where

$$f(u) = \frac{1}{(1-u)^{\theta}}. \tag{2.2}$$

θ will be set to 1 for the duration of this paper. The boundary conditions here are the same as those used in [9], therefore $u(t,0) = u(t,1) = 0$. However, they did not specify an initial condition in their paper. As such, with the one dimensional adaptive method in mind (this will come clear later), the initial condition applied will be $u(0,x) = \frac{1}{10}\sin(\pi x)$.

Now that the initial and boundary conditions have been set, the solution can begin. If the mesh indices are j, n and one lets $u(t,x) = u(n\Delta t, j\Delta x) \approx u_j^n$ and then expands the u_{xx} terms using a central difference scheme and the u_t term as a forward difference scheme, then (2.1) becomes

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{a^2 \Delta x^2} + f(u_j^n). \tag{2.3}$$

which can be rearranged to show that

$$u_j^{n+1} = u_j^n + \Delta t \left[\frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{a^2 \Delta x^2} + f(u_j^n) \right]. \tag{2.4}$$

By considering (2.1) and using the fact that an approximation for u has been found using (2.4), one can also very quickly derive a approximation for u_t. If the right hand side of equation (2.1) is expanded in the same way as above, then one has

$$(u_t)_i = \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{a^2 \Delta x^2} + f(u_j^n). \tag{2.5}$$

Having an approximation for u_t as well as u is particularly useful, as it is the u_t value's growth which causes the quenching and as such, an approximation of it with a known truncation error provides a useful tool for analysing the adaptive one dimensional case.

Using this fairly straightforward method of numerically solving the problem also allows for the truncation error of the scheme to be found fairly simply. If we consider equation (2.3), then the truncation error estimate can be calculated by first considering

$$\tau_j^n = \frac{u_j^{n+1} - u_j^n}{\Delta t} - \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{a^2 \Delta x^2} - f(u_j^n), \tag{2.6}$$

with u_j^n replaced by $u(j\Delta x, n\Delta t)$. These terms can all be expanded using the *Taylor Series* about $u(j\Delta x, n\Delta t)$. Applying this expansion to equation (2.6) results in

$$\tau_j^n = \tfrac{1}{\Delta t}\left[u + \Delta t u_t + \tfrac{\Delta t^2}{2} u_{tt} + \cdots - u\right]$$

$$-\tfrac{1}{a^2 \Delta x^2}\left[u - \Delta x u_x + \tfrac{\Delta x^2}{2} u_{xx} - \tfrac{\Delta x^3}{3!} u_{xxx} + \tfrac{\Delta x^4}{4!} u_{xxxx} + \dots\right.$$

$$\left. -2u + u + \Delta x u_x + \tfrac{\Delta x^2}{2} u_{xx} + \tfrac{\Delta x^3}{3!} u_{xxx} + \tfrac{\Delta x^4}{4!} u_{xxxx} + \dots\right] - f(u),$$

where u represents $u(j\Delta x, n\Delta t)$ here. Clearly, many of these terms cancel, which leaves us with

$$\tau_n = u_t + \frac{\Delta t}{2} u_{tt} - \frac{1}{a^2} u_{xx} - \frac{\Delta x^2}{12} u_{xxxx} - f(u) + O(\Delta t^2) + O(\Delta x^4).$$

Using equation (2.1), three more terms can be removed, thus reducing the above equation to

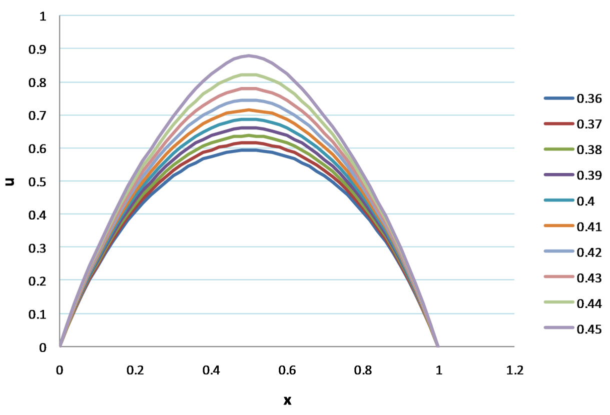
$$\tau_n = \frac{\Delta t}{2} u_{tt} - \frac{\Delta x^2}{12} u_{xxxx} + O(\Delta t^2) + O(\Delta x^4). \tag{2.7}$$

Therefore the method is first order in time and second order in space.

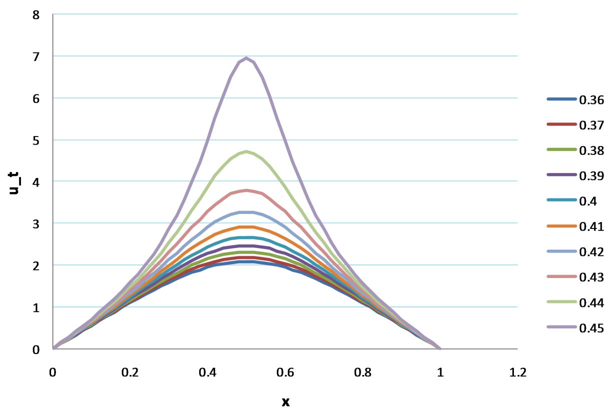
Applying the schemes from equations (2.4) and (2.5), using different time steps and numbers of nodes, produces reasonable results even using time steps as large as 0.01 and Δx as large as 0.1. However, in an attempt to keep the errors low and since this is the simplest method being used, Δt has been set to 0.0001 and Δx to 0.02. This should give a reasonable estimate as to when the problem quenches under these conditions.

Applying these conditions results in the quenching occurring at t=0.4588. Using a Δt value of 0.0001 finds a convergent result. Lower values find a more exacting quenching point, but are in the same region as produced using these conditions. It is known from both [9] and [7] that this problem is going to quench at the centre of the domain, in finite time, provided that the domain $(0, a)$ is larger than $2\sqrt{2}$. Since a is set to π throughout this study, the problem will quench in finite time and at the centre of the domain.

The figures below illustrate the behaviour of u and u_t in time.

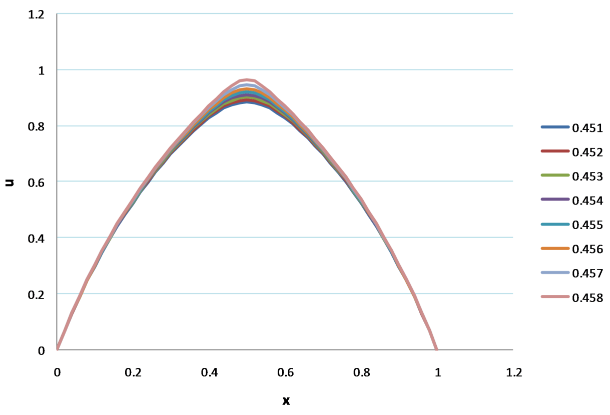


(a) u plotted against x

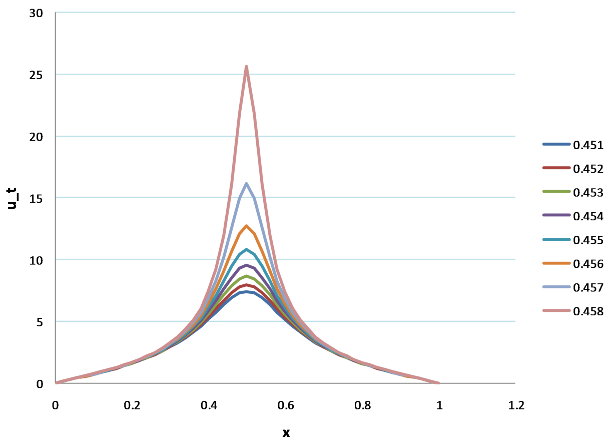


(b) u_t plotted against x

Figure 2.1: u and u_t plotted in time between $t = 0.36$ and $t = 0.45$ with intervals of 0.01.

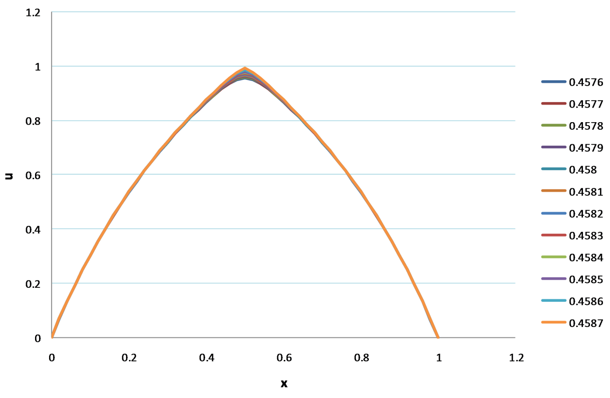


(a) u plotted against x

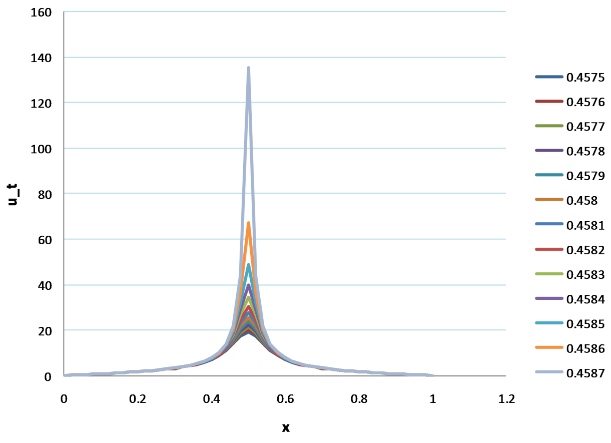


(b) u_t plotted against x

Figure 2.2: u and u_t plotted in time between $t = 0.451$ and $t = 0.458$ with intervals of 0.001.



(a) u plotted against x



(b) u_t plotted against x

Figure 2.3: u and u_t plotted in time between $t = 0.4575$ and $t = 0.4587$ with intervals of 0.0001.

We can see from the results in figures 2.1-2.3 results that, given the parameters $\Delta t = 0.0001$ and $\Delta x = 0.02$, the problem quenches at approximately $t = 0.4587$. Figure 2.3a shows how u is just touching 1 at this time and by this point u_t has certainly blown up, as evidenced by figure 2.3b. By the next time step the solution is post quenching and the results no longer provide any useful information.

These results correspond to those of [9] and the solution behaves as stated by [7]. As such, these results should give a reasonable estimation of what to expect from the one dimensional adaptive case, as this will use the same initial and boundary conditions.

2.2 A Test in Two Dimensions

In this two dimensional case, equation (2.1) will need to be modified so that it applies to a two dimensional case. The problem is known to be symmetrical and therefore, the problem will also have radial symmetry. This lends it to being solved as a radial differentially equation, rather than using a $\underline{\nabla}^2 u$ term. Therefore, as a radial problem, equation (2.1) becomes

$$u_t = \frac{1}{a^2 r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) + f(u), \tag{2.8}$$

where $f(u)$ is still defined as in equation (2.2), with θ remaining set to one.

If $u(t,r) = u(n\Delta t, jh) \approx u_j^n$ (where h is the spacing between nodes along the radius) and both the space and time derivatives are expanded as forward differences, then equation (2.8) becomes

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{1}{ha^2 r_j}\left(r_{j+1/2}\left(\frac{u_{j+1}^n - u_j^n}{h}\right) - r_{j-1/2}\left(\frac{u_j^n - u_{j-1}^n}{h}\right)\right) + f(u_j^n),$$

where $r_{j+1/2} = \frac{1}{2}\left(r_{j+1} + r_j\right)$. Rearranging this shows that

$$u_j^{n+1} = u_j^n + \frac{\Delta t}{h^2 a^2 r_j}\left(r_{j+1/2}\left(u_{j+1}^n - u_j^n\right) - r_{j-1/2}\left(u_j^n - u_{j-1}^n\right)\right) + f(u_j^n)\Delta t. \tag{2.9}$$

Once again, u_t can be approximated once the new u values have been calculated, this time using

$$(u_t)_j^n = \frac{1}{ha^2 r}\left(r_{j+1/2}\left(\frac{u_{j+1}^n - u_j^n}{h}\right) - r_{j-1/2}\left(\frac{u_j^n - u_{j-1}^n}{h}\right)\right) + f(u_j^n). \tag{2.10}$$

However, this method does require a slight alteration to work throughout the domain. If we look at the above scheme, it can been seen that there is a division by the radial position r_j. This is not a problem throughout the domain except for one point. At the origin, this value is 0 and thus causes a singularity. As such, an alteration is made using using the substitution proposed by Smith [10] (p76). If for the solution of u_t at the origin we substitute in $2u_{rr}$ for $\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)$, then (2.8) becomes

$$u_t = \frac{1}{a^2}2u_{rr} + f(u).$$

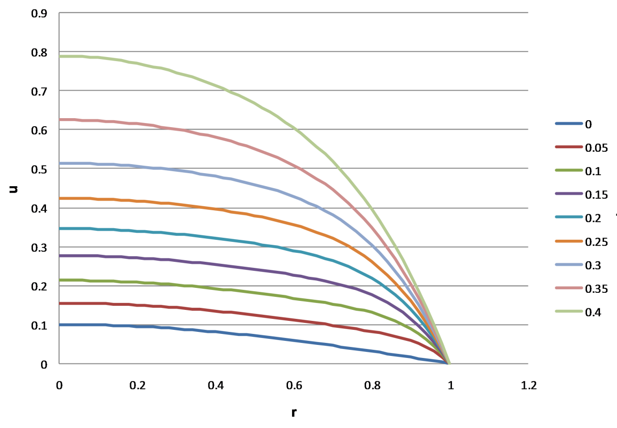
Applying a central difference schemes to this results in

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}=\frac{2}{h^2a^2}\left(u_{j-1}^n-2u_j^n+u_{j+1}^n\right)+f(u_j^n),$$

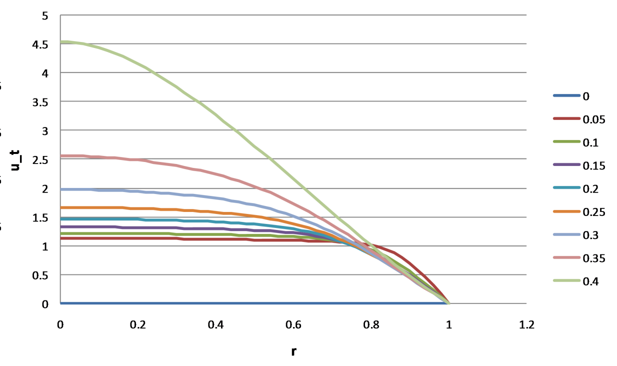
which can be further rearranged to show that

$$u_j^{n+1}=\frac{2\Delta t}{h^2a^2}\left(u_{j-1}^n-2u_j^n+u_{j+1}^n\right)+\Delta t f(u_j^n)+u_j^n. \tag{2.11}$$

With this condition imposed and the initial condition $u(0,r)=\frac{1}{10}\cos\left(\frac{\pi}{2}r\right)$ and the boundary condition $u(t,1)=0$, the solution can begin. The following figures show the solution using $\Delta t=0.0001$ and $\Delta x=0.02$ for several different time intervals.

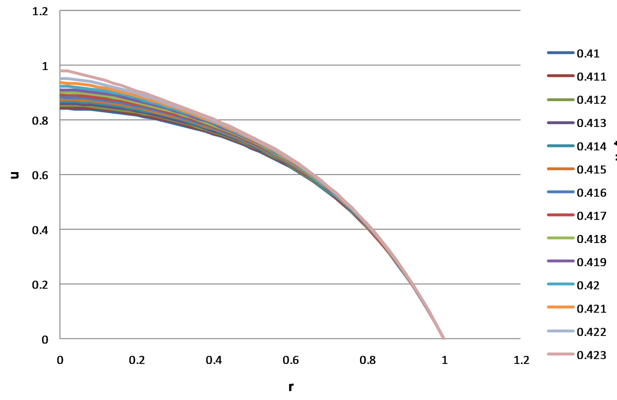


(a) u plotted against r

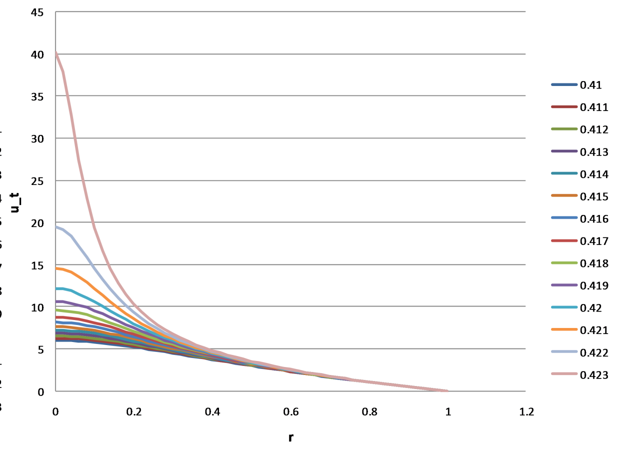


(b) u_t plotted against r

Figure 2.4: u and u_t, plotted in time between $t=0$ and $t=0.4$ with intervals of 0.05.

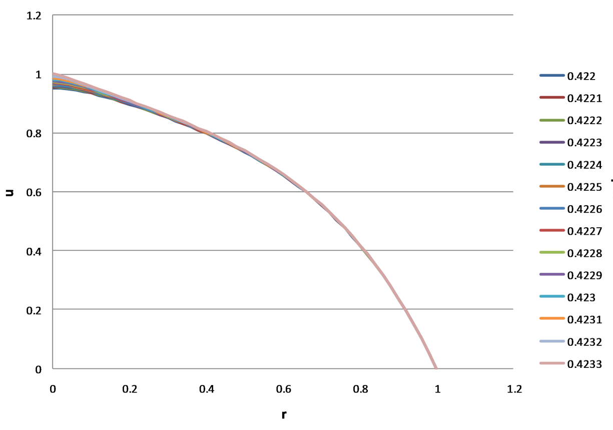


(a) u plotted against r

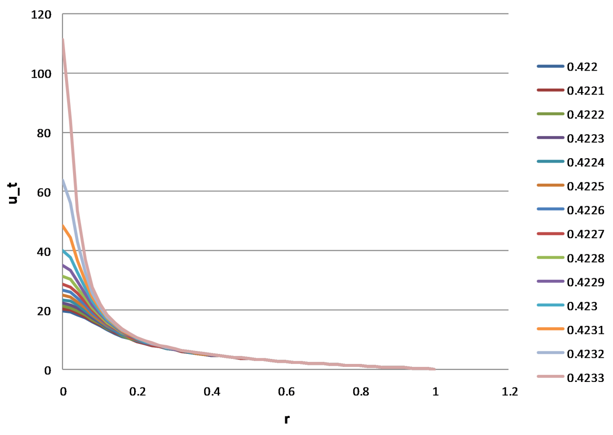


(b) u_t plotted against r

Figure 2.5: u and u_t, plotted in time between $t=0.41$ and $t=0.423$ with intervals of 0.001.



(a) u plotted against r



(b) u_t plotted against r

Figure 2.6: u and u_t, plotted in time between $t = 0.422$ and $t = 0.4233$ with intervals of 0.0001.

This version of the problem quenches at $t = 0.4234$, which is slightly faster the one dimensional case, although it follows the expected pattern as u and u_t evolve, and finally it quenches on the left hand boundary, which represents the origin here. As with the one dimensional case, the Δt and Δx chosen ensure that the problem is converging. Using smaller values would show a more accurate solution, but it would still quench at around the same point in time.

Two preliminary sets of results have now been produced. They show what an approximate result should be when using the two different schemes for the two cases and will later be used as a comparison with the results from the adaptive methods. If the adaptive methods are working correctly, one would expect them to quench at a time similar to that of these methods.

Chapter 3

One Dimensional Adaptive Method

Starting from the scaled equation from [9], we have

$$u_t = \frac{1}{a^2} u_{xx} + f(u) \tag{3.1}$$

where $f(u) = \frac{1}{(1-u)^\theta}$ and $\theta = 1$. The boundary conditions are given by $u(0,t) = u(1,t) = 0$ and the initial condition chosen is $u(x,0) = \frac{1}{10} sin(\pi x)$. This choice of initial condition will be explained later.

There are numerous ways in which to apply an adaptive method. In this case and using this equation, it is known that the u_t term in the solution of the PDE is the part of the equation causing the quenching or blow-up. In [1] u is used as the monitor function, but in this dissertation u_t will be used to govern the refinement of the mesh in a very similar fashion to that of [1]. If one lets A_i be the absolute area between two points x_i and x_{i+1} then we have

$$\int_{x_i}^{x_{i+1}} u_t \, dx = A_i. \tag{3.2}$$

However, since the total integral from 0 to 1 is not conserved, conservation of the absolute area is of little use in this case. A better measure is that of a percentage or relative area with respect to the total area under u_t. If we define a new variable σ to represent the total area and C_i to represent the relative area under the curve between x_i and x_{i+1} then we have

$$\sigma = \int_0^1 u_t \, dx \tag{3.3}$$

and the relative area is

$$C_i = \frac{1}{\sigma} A_i = \frac{1}{\sigma} \int_{x_i}^{x_{i+1}} u_t \, dx. \tag{3.4}$$

It is this quantity that will be conserved in time. At this point the choice of initial conditions must be explained. From [9], we know that the initial u values must be in the region $0 \leq u_0 < 1$, with $u(0) = u(1) = 0$). Also, equation (3.3) stipulates that the initial condition must also be twice differentiable in time, due to the use of the initial u_{xx} term. The sine function $\frac{1}{10} \sin(\pi x)$ is used, since it satisfies these criteria.

We need to know how σ changes with respect to time, as this will affect the areas C_i in equation (3.4). As such, we differentiate equation (3.3) with respect to time to give

$$\frac{d\sigma}{dt} = \frac{d}{dt} \int_0^1 u_t \, dx = \int_0^1 u_{tt} \, dx.$$

We can now substitute equation (3.1) into this to give

$$\frac{d\sigma}{dt} = \int_0^1 (\frac{1}{a^2} u_{xx} + f(u))_t \, dx.$$

Applying the time derivative and using the fact that the differentiations can be carried out in any order produces

$$\frac{d\sigma}{dt} = \int_0^1 \frac{1}{a^2} (u_t)_{xx} + f'(u) u_t \, dx.$$

From equation (3.1), u_t can be substituted in again to give

$$\frac{d\sigma}{dt} = \int_0^1 \frac{1}{a^2} \left(\frac{1}{a^2} u_{xx} + f(u) \right)_{xx} + f'(u) \left(\frac{1}{a^2} u_{xx} + f(u) \right) dx.$$

Integrating gives

$$\dot{\sigma} = \frac{1}{a^4} u_{xxx} \bigg|_0^1 + \frac{1}{a^2} f'(u) u_x \bigg|_0^1 + \int_0^1 f'(u) \left(\frac{1}{a^2} u_{xx} + f(u) \right) dx, \tag{3.5}$$

where $\dot{\sigma} = \frac{d\sigma}{dt}$. This equation will be considered later, when the numerical solution begins.

Now, equation (3.4) states that the relative area under u_t is constant, so differentiating it with respect to time gives

$$\frac{d}{dt} \left[\frac{1}{\sigma(t)} \int_{x_i(t)}^{x_{i+1}(t)} u_t(t) \, dx \right] = 0.$$

Since σ and u_t are both functions of t, the product rule must first be applied to give

$$\frac{d}{dt} \left(\frac{1}{\sigma} \right) \int_{x_i}^{x_{i+1}} u_t \, dx + \frac{1}{\sigma} \frac{d}{dt} \int_{x_i}^{x_{i+1}} u_t \, dx.$$

If we firstly considers the second term, then we can see that the *Leibniz Integral Rule* can be applied to show that

$$\frac{d}{dt}\int_{x_i}^{x_{i+1}} u_t\,dx = \int_{x_i}^{x_{i+1}} u_{tt}\,dx + \left[u_t\frac{dx}{dt}\right]_{x_i}^{x_{i+1}},$$

which implies that

$$\frac{d}{dt}\left(\frac{1}{\sigma}\right)\int_{x_i}^{x_{i+1}} u_t\,dx + \frac{1}{\sigma}\left(\int_{x_i}^{x_{i+1}} u_{tt}\,dx + \left[u_t\frac{dx}{dt}\right]_{x_i}^{x_{i+1}}\right) = 0.$$

Once again, u_t can be substituted into the u_{tt} term from equation (3.1), as well as using $\frac{d}{dt}(\frac{1}{\sigma}) = -\frac{\dot{\sigma}}{\sigma^2}$. This gives

$$-\frac{\dot{\sigma}}{\sigma^2}\int_{x_i}^{x_{i+1}} u_t\,dx + \frac{1}{\sigma}\left(\int_{x_i}^{x_{i+1}} \left(\frac{1}{a^2}u_{xx} + f(u)\right)_t\,dx + \left[u_t\frac{dx}{dt}\right]_{x_i}^{x_{i+1}}\right) = 0$$

Clearly this can be multiplied through by σ, but also note that earlier C_i was defined as equation (3.4). This can be substituted into the first term, along with the multiplication by σ to give

$$-\dot{\sigma}C_i + \int_{x_i}^{x_{i+1}} \left(\frac{1}{a^2}u_{xx} + f(u)\right)_t\,dx + \left[u_t\frac{dx}{dt}\right]_{x_i}^{x_{i+1}} = 0.$$

Substituting equation (3.1) in again and rearranging produces

$$\left[u_t\frac{dx}{dt}\right]_{x_i}^{x_{i+1}} = \dot{\sigma}C_i - \frac{1}{a^2}\int_{x_i}^{x_{i+1}} \left(\frac{1}{a^2}u_{xx} + f(u)\right)_{xx}\,dx - \int_{x_i}^{x_{i+1}} \frac{1}{a^2}f'(u)u_t\,dx.$$

A little more simplification results in

$$\left[u_t\frac{dx}{dt}\right]_{x_i}^{x_{i+1}} = \dot{\sigma}C_i - \frac{1}{a^4}u_{xxx}\Big|_{x_i}^{x_{i+1}} - \frac{1}{a^2}f'(u)u_x\Big|_{x_i}^{x_{i+1}} - \int_{x_i}^{x_{i+1}} \frac{1}{a^2}f'(u)u_t\,dx. \tag{3.6}$$

The majority of these terms are known, although not all of them are simple to calculate. The integral can be approximated using the trapezium rule and the first derivatives by standard finite differences. However, the third derivative u_{xxx} is more complicated and will be evaluated using finite elements.

3.1 The Finite Element Part

To evaluate the u_{xxx} term in equation (3.6), one can use a finite element method. If one begins by introducing $v = -u_{xx}$, then a weak form is

$$\int_0^1 \phi_i v\,dx = -\int_0^1 \phi_i u_{xx}\,dx,$$

where ϕ_i is a one dimensional linear hat function. Subsequently, integration by parts shows that

$$\int_0^1 \phi_i v \, dx = -\phi_i u_x \mid_0^1 + \int_0^1 (\phi_i)_x u_x \, dx.$$

This can be written in matrix form as

$$M\underline{v} = \underline{\phi} + K\underline{u}, \tag{3.7}$$

where $\underline{\phi} = (u_x|_0, 0, 0, ..., 0, -u_x|_N)$,

$$M = \begin{pmatrix} \frac{x_1-x_0}{3} & \frac{x_1-x_0}{6} & 0 & \ddots & \ddots \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & \frac{x_i-x_{i-1}}{6} & \frac{x_{i+1}-x_{i-1}}{3} & \frac{x_{i+1}-x_i}{6} & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ \ddots & \ddots & 0 & \frac{x_N-x_{N-1}}{6} & \frac{x_N-x_{N-1}}{3} \end{pmatrix}$$

and

$$K = \begin{pmatrix} \frac{1}{x_1-x_0} & -\frac{1}{x_1-x_0} & 0 & \ddots & \ddots \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & -\frac{1}{x_i-x_{i-1}} & \frac{1}{x_i-x_{i-1}} + \frac{1}{x_{i+1}-x_i} & -\frac{1}{x_{i+1}-x_i} & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ \ddots & \ddots & 0 & -\frac{1}{x_N-x_{N-1}} & \frac{1}{x_N-x_{N-1}} \end{pmatrix}.$$

At this point, $\underline{u}$ and $\underline{x}$ are known and thus the solution for $\underline{v}$ can be found by applying a tri-diagonal matrix solver, in this case the Thomas algorithm.

3.2 Numerical Solution

The first part of the numerical solution is to calculate the v values. The explanation for calculating these values is given above. However, the initial v values can be found in a far simpler manner. It is known that $v = -u_{xx}$ and that when $t = 0$, $u = \frac{1}{10} sin(\pi x)$. Therefore the initial values of v are given by $v = \frac{\pi^2}{10} sin\,(\pi x)$. At later times, v is calculated using the method explained in the previous section.

Once the initial v values have been calculated the C_i values can be found from equation (3.4). Substituting equation (3.1) into (3.4) gives

$$C_i = \frac{1}{\sigma} \int_{x_i}^{x_{i+1}} \frac{1}{a^2} u_{xx} + f(u) \, dx.$$

Substituting $v = -u_{xx}$ into this gives

$$C_i = \frac{1}{\sigma} \int_{x_i}^{x_{i+1}} f(u) - \frac{v}{a^2}\, dx, \tag{3.8}$$

which can be approximated using the trapezium rule. The C_i values are only calculated once, directly after the initial $v's$ have been calculated and are constant for all time.

If we first consider the problem at $t = 0$, we know the exact values of u at all points. Once the initial values have been applied, the v values can be calculated and thus an approximation to u_t can also be calculated. This is given by

$$u_t = f(u) - \frac{v}{a^2}. \tag{3.9}$$

Once the time loop begins, u_t is considered in two different ways. Firstly an approximation of u_t is calculated from equation (3.9), using v at each node. All of the other calculations mentioned so far use this approximation. u_t is approximated at the end of the loop, this time using a midpoint rule applied to equation (3.4). The v values are also recalculated at the end of the loop.

Now that u_t has a value at all points, the time loop can begin. Bearing in mind that u_t, σ, the C_i values and $\underline{v}$ have all been calculated before the time loop begins, the first thing to be calculated in the loop is $\dot{\sigma}$.

$\dot{\sigma}$ has already been defined in equation (3.5). However, in equation (3.5), all of the u_t terms have been expanded using equation (3.1), but now that u_t has been approximated using v, it can be reintroduced, since this is far simpler to calculate than the two terms which are otherwise created. As such, equation (3.5) becomes

$$\dot{\sigma} = \left.\frac{u_{tx}}{a^2}\right|_0^1 + \int_0^1 f'(u) u_t\, dx. \tag{3.10}$$

The first term can be approximated using finite differences applied to u_t, while the integral can be approximated using the composite trapezium rule. The approximation to the u_{tx} term is given by one sided differences, as in

$$\left.\frac{u_{tx}}{a^2}\right|_0^1 \approx \frac{1}{a^2}\left(\frac{(u_t)_N - (u_t)_{N-1}}{x_N - x_{N-1}} - \frac{(u_t)_1 - (u_t)_0}{x_1 - x_0} \right)$$

Once $\dot{\sigma}$ is known, the next σ value in time is calculated using the forward Euler method

$$\sigma^{n+1} = \sigma^n + \Delta t \dot{\sigma}$$

with a suitable choice of Δt. Next comes the movement of the nodes. This is calculated using equation (3.6). In equation (3.6), every term is known apart from the two $\frac{dx}{dt}$ terms. However, two of the $\frac{dx}{dt}$ values are known. To keep the domain the same size for all time, the two outermost nodes (x_0 and x_N) are fixed in position and therefore $\frac{dx}{dt}$ at these nodes is 0. In equation (3.6) the $\frac{dx}{dt}$ terms are evaluated at x_i and x_{i+1}. Starting from $i = 0$, the evaluation at x_i (since there is no node movement at this point) is known and therefore equation (3.6) can be rearranged to find the unknown $\frac{dx}{dt}$ term and all the subsequent $\dot{x}$ terms explicitly.

As with equation (3.5), one can reform any expanded u_t terms using (3.1), as u_t is now known. As such, equation (3.6) becomes

$$\left[u_t \frac{dx}{dt}\right]_{x_i}^{x_{i+1}} = \dot{\sigma} C_i - \left.\frac{u_{tx}}{a^2}\right|_{x_i}^{x_{i+1}} - \int_{x_i}^{x_{i+1}} \frac{1}{a^2} f'(u) u_t \, dx.$$

This can then be rearranged into the form

$$\dot{x}_{i+1} = \frac{\alpha - \beta - \gamma}{u_t|_{i+1}}, \tag{3.11}$$

where

$$\dot{x}_{i+1} = \left.\frac{dx}{dt}\right|_{x_{i+1}}$$

$$\alpha = \dot{\sigma} C_i + u_t \dot{x}|_{x_i} \tag{3.12}$$

$$\beta = \left.\frac{u_{tx}}{a^2}\right|_{x_i}^{x_{i+1}} \tag{3.13}$$

and

$$\gamma = \int_{x_i}^{x_{i+1}} f'(u) u_t \, dx. \tag{3.14}$$

β can generally be approximated using finite differences, with the derivatives being approximated using a central difference scheme. However, the scheme for calculating β does need to be slightly modified if the velocity being calculated is at the first internal node x_1.

Therefore in general

$$\beta \approx \frac{1}{a^2}\left(\frac{(u_t)_{i+2} - (u_t)_i}{x_{i+2} - x_i} - \frac{(u_t)_{i+1} - (u_t)_{i-1}}{x_{i+1} - x_{i-1}}\right). \tag{3.15}$$

But, at $i = 0$

$$\beta \approx \frac{1}{a^2}\left(\frac{(u_t)_{i+2} - (u_t)_i}{x_{i+2} - x_i} - \frac{(u_t)_{i+1} - (u_t)_i}{x_{i+1} - x_i}\right).$$

γ can be approximated by simply using the trapezium rule. However, wherever the trapezium rule is being used to approximate an integral between x_i and x_{i+1}, the approximation accuracy is very dependent on the number of nodes being used. The values between nodes are unknown and thus it is not possible to increase the number of trapeziums being used to approximate any given integral. Therefore to maintain the accuracy of these approximations, a reasonable number of nodes must be used throughout.

Once these velocities have been calculated, the positions of the nodes can be updated using the same forward Euler method,

$$x_i^{n+1} = x_i^n + \Delta t \dot{x}_i^n. \tag{3.16}$$

Once the nodes have been repositioned, the new value of u_t at the nodes can be approximated by applying the midpoint rule to equation (3.4) in the form

$$\frac{1}{\sigma}\int_{i-1}^{i+1} u\,dx = C_{i-1} + C_i.$$

The standard midpoint rule applied here produces

$$(x_{i+1} - x_{i-1})\, u_t\big|_{x_i} \approx \sigma(C_{i-1} + C_i),$$

which can then be rearranged to show that

$$u_t\big|_{x_i} \approx \sigma\frac{C_{i-1} + C_i}{x_{i+1} - x_{i-1}}. \tag{3.17}$$

The extra σ is present to make up for the fact that the C's are actually a relative measurement of the areas under u_t, rather than the absolute values. It would be possible to use the absolute areas A_i, but this would mean recalculating A_i at every time step using the trapezium rule, rather than applying the above, simpler step.

Normally (if a fixed grid is used), since u_t is now known, one could use

$$u_i^{n+1} = u_i^n + \Delta t u_t$$

to find the new u_i values. However, a different approach must be taken here. We start by defining a new variable Θ to be

$$\Theta_i = \int_{x_{i-1}}^{x_{i+1}} u\,dx. \tag{3.18}$$

Θ_i can be approximated using a trapezium rule before the time loop begins. We can then differentiate Θ_i and use this derivative to calculate the new Θ_i values. As such, differentiating (3.18) with respect to time produces

$$\dot{\Theta}_i = \frac{d}{dt}\int_{x_{i-1}}^{x_{x+1}} u\,dx.$$

Applying the *Leibniz Integral Rule* to this results in

$$\dot{\Theta}_i = \int_{x_{i-1}}^{x_{x+1}} u_t\,dx + [u\dot{x}]_{x_{i-1}}^{x_{i+1}}. \tag{3.19}$$

The first term in equation (3.19) can be calculated using a composite trapezium rule (the u_t value here is the midpoint approximation) once again, while the second term is a straightforward operation, since $\underline{\dot{x}}$ has already been found.

Now that $\dot{\Theta}_i$ is known, the new Θ_i values can simply be calculated using $\Theta_i^{n+1} = \Theta_i^n + \Delta t\dot{\Theta}_i$. Once the new values of Θ are known, the new u values can be calculated using a midpoint rule. This produces

$$u_i = \frac{\Theta_i}{x_{i+1} - x_{i-1}} \tag{3.20}$$

With the new u values known, the v and u_t values are recalculated, the time step advanced and then the time loop begins again. A summary of the numerical method in terms of its pseudo code is displayed in Appendix 1.

3.3 Results

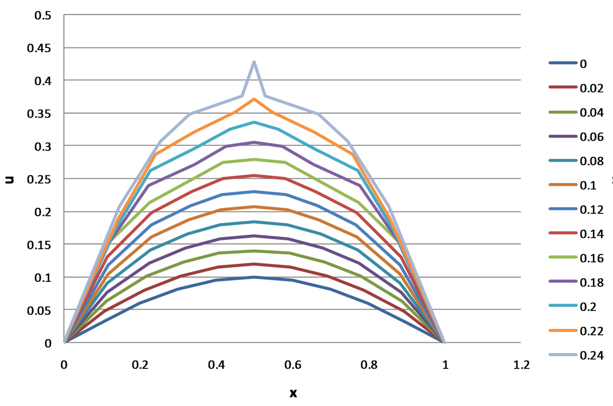
The method explained was coded up in C++ but does not seem to produce any form of usable results, as the scheme becomes hugely unstable very quickly. The method does not last anywhere near as long as the preliminary results would suggest, which seems to be a problem caused by the movement of the nodes. Quite quickly certain nodes begin to overlap and thus tangle up the mesh. As such, a number of adaptations have been considered in an attempt to allow the method to produce a result similar to that of the preliminary test.

3.4 Adaptations

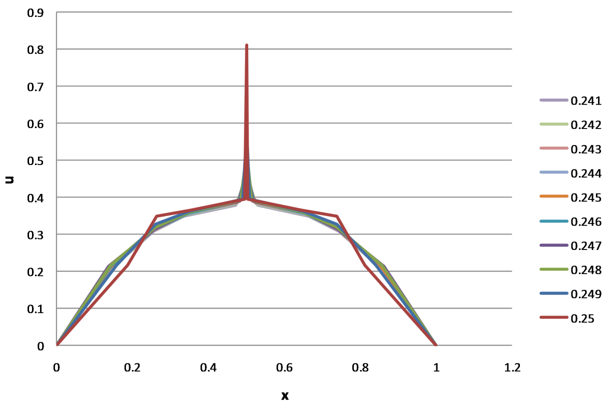
The first adaptation used is a simple one. If we consider the calculation of β in equation (3.15), we can see that generally β can be calculated using a central difference scheme. However, when considering β when calculating the velocity of the first internal point, a forward difference scheme must be considered, since the boundary conditions do not allow for any form of ghost point to be considered. This is a problem which can be mediated using the symmetry of the problem. It is known from both [8] and [9] that equation (3.1) is symmetric around the centre of the domain. Therefore we can calculate the the velocity of the nodes from the centre to $u = 1$ (where $\dot{x} = 0$ and then use those values to state the velocities of their opposite nodes on the other side of the centre node.

This however, does enforce another condition on the problem, which is that as well as $\dot{x}$ being 0 at both $x = 0$ and $x = 1$, $\dot{x}$ must also be 0 at $x = 0.5$. This further implies that there must be an odd number of nodes so that one node lies directly on $x = 0.5$.

Applying these conditions does allow the method to produce more sensible results, although they are still not entirely expected. The method is far more unstable than the fixed mesh method, but it will run for slightly longer than the original adaptive method using $N = 10$ (N represents the number of areas created, so the number nodes is in fact $N + 1$) and a Δt value of 0.001, but still fails very quickly. However, using a smaller time step or a greater number of nodes results in the method failing more quickly, so the plots below use $\Delta t = 0.001$ and $\Delta x = 0.1$.



(a) The solution from $t = 0$ to $t = 0.24$ at intervals of 0.02.

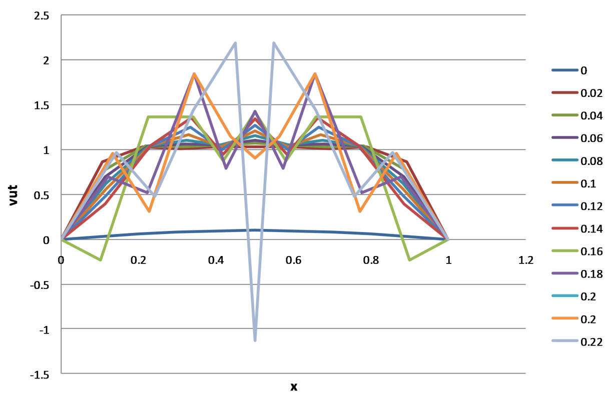


(b) The solution from $t = 0.241$ to $t = 0.25$ at intervals of 0.001.

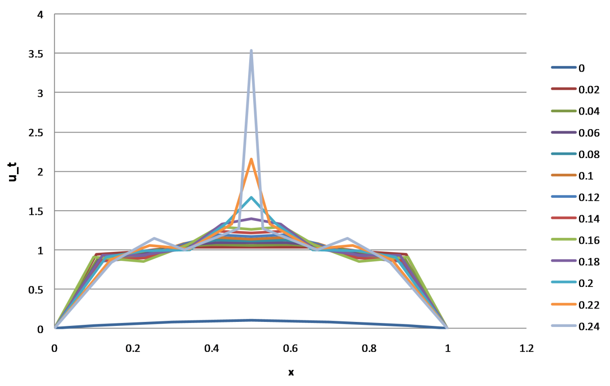
Figure 3.1: u plotted against x using $\Delta t = 0.001$ and $\Delta x = 0.1$

So, using this slight modification (forcing the central node to remain stationary), a quenching solution can be found. However, it is clearly not forming a solution overly similar to that of the preliminary tests. It lasts a little over half of the time that the preliminary method could run for under the same conditions before quenching. Although whether this can even be described as quenching is debatable. It should also be noted that adding more nodes and reducing Δt did not improve this solution. Reducing the time step makes little to no difference, while adding more nodes forces the method to fail even faster. Lowering the number of nodes does appear to aid stability, but the solution becomes unusable in the process as it becomes near triangular.

A greater difference between the methods so far can been when comparing the u_t from the preliminary method with that of the u_t values calculated using v and the midpoint rule from the adaptive method.



(a) The v estimate of u_t.



(b) The midpoint rule version of u_t

Figure 3.2: The two different methods of calculating u_t in the time period of $t = 0$ to $t = 0.24$, with intervals of $\Delta x = 0.1$ and a Δt value of 0.001.

Other adaptations were made, such as forcing the v end point values to be equal to a^2 (by rearranging (3.1)) and attempting to solve for v using a central finite difference scheme instead of a finite element scheme. The reasoning behind trying to use these two methods is that in the formulation of the finite elements solution to v, the hat functions (ϕ) and therefore u_x at each end of v remain present. These hat functions themselves are not particularly an issue, it is the multiplication by the u_x term evaluated on the boundary which is the problem.

The only boundary conditions given are that u (and therefore u_t) must be equal to zero at the boundaries, which means that u_x is not specified. This poses a problem as an estimation of u_x needs to be made. A simple way to try to find these terms was to apply a forward finite difference scheme to the u_x term at $x = 0$ and a backwards difference at $x = 1$. However, this is very crude and likely to be a major cause of problems during the solution of v. Unfortunately, without a boundary condition, these terms are very difficult to evaluate.

Based on this, another adaptation considered was assuming that $u_x = 0$ weakly on the boundaries alongside $u = 0$, the aim being to remove the extra terms created in the calculations of v. However, this proved even more unstable than having the approximation to u_x in the solution.

On a positive note, forcing the central node to remain in the same position throughout and thus removing the tricky β term does appear successful. The β term when calculating the velocity of the first internal point also needed to be considered using a forward difference scheme. When solved using this, the method barely worked at all. Removing that term by using the symmetry of the problem seemingly improved it. This in turn suggests that perhaps the forward and backward difference schemes used in the calculation of v is another element of the solution causing an issue.

3.5 Critical Analysis

Clearly this method has proved somewhat unsuccessful in this study. Admittedly, it could be something as simple as a programming error, although this seems unlikely. Using the method as originally derived proved completely unusable. The introduction of the forced central node seems to help, but the solution produced is still very poor. Even with the addition of the stationary central node, the scheme is highly unstable, especially when altering the number of nodes. Raising N above 10 results in the solution method failing very quickly indeed, due to the unstable nature of forward Euler time stepping.

It is possible that the issue is inherent in the boundary conditions. By forcing u to be zero at the boundaries, it is possible that u_t suffers at the boundaries since u_t contains a u_{xx} term, which at the boundary is going to differ greatly from that of the internal points because u is being forced to 0. Now, since the monitor function here is u_t, it stands to reason that perhaps it is the boundary conditions causing the issue, since they are forcing u_t to have erratic values near the boundaries. One can see in figure 3.2b that there are three areas with large gradients in the v approximation to u_t. These are most likely caused by the large u_{xx} values being found nearest the boundaries.

One could also consider that the choice of monitor function is not ideal. It can be viewed in two ways. [9] stated that it is u_t causing the blow up and thus, while hoping to improve the resolution of the mesh about the quenching point, it could be argued that moving the mesh with regards to the term forcing the blow up may work. Conversely, since it is u_t is causing the blow up, it could also be suggested that this is likely to force the mesh to tangle as u_t tends to infinity.

Chapter 4

A Two Dimensional Adaptive Method

Like the one dimensional case this method begins with the equation from [9], but in this case it clearly needs to be modified slightly. The equation has already been considered once as a radial problem, but here it will be considered in two dimensions and thus becomes

$$u_t = \frac{1}{a^2}\nabla^2 u + f(u). \tag{4.1}$$

For this case, $f(u)$ remains as before ($f(u) = \frac{1}{(1-u)^\theta}$, with θ set to 1), but the initial conditions are altered slightly. In this case the domain is circular and although the problem will be solved using radial symmetry and thus the solution will appear one dimensional, the same initial conditions cannot be used to represent each end of the radius. Instead, the outer boundary of the domain will be taken to be 0 and the origin will have the condition $\underline{\nabla} u = 0$ imposed upon it.

Like the one dimensional case, a twice differentiable function is required in the domain which satisfies the boundary conditions. In particular $u(x, y, 0) = \frac{1}{10}\cos\left(\frac{2}{\pi}\sqrt{x^2+y^2}\right)$ will be used.

As in the one dimensional case, the means by which the points move centres around conserving the relative volumes between points with respect to u_t. So in this case we can let

$$\sigma(t) = \int_{\Omega(t)} u_t \, d\Omega, \tag{4.2}$$

where Ω represents the domain on which the equation (equation (4.1)) is being solved, and consider the relative area for this method, thus defining constants c_i which are given by

$$c_i = \frac{1}{\sigma}\int_{\Omega_i} u_t \, d\Omega. \tag{4.3}$$

The integral over Ω_i here represents the integral over the area created between two circular rings of radius r_i and r_{i+1}. If equation (4.3) is rearranged, we get

$$\sigma c_i = \int_{\Omega_i} u_t \, d\Omega. \tag{4.4}$$

However, once the mesh begins its movement the σ values will be calculated using a finite element method. This entails introducing a test (a member of a partition of unity) function to equation (4.3) to produce

$$c_i = \frac{1}{\sigma}\int_{\Omega} w_i u_t \, d\Omega, \tag{4.5}$$

where w moves with $\underline{v}$ and therefore must satisfy the advection equation

$$\frac{\partial w_i}{\partial t} + \underline{v} \cdot \underline{\nabla} w_i = 0. \tag{4.6}$$

Equation (4.5) can then be rearranged to show that

$$\sigma c_i = \int_{\Omega} w_i u_t \, d\Omega. \tag{4.7}$$

For the initial σ values, all the u terms are known initially and the rings are evenly spaced. Now, we need to know how σ changes with time and therefore equation (4.7) must be differentiated with respect to time. This gives

$$\dot{\sigma} c_i = \frac{d}{dt}\int_{\Omega} w_i u_t \, d\Omega,$$

where $\dot{\sigma} = \frac{d\sigma}{dt}$. Using the *Reynolds Transport Theorem* this becomes

$$\dot{\sigma} c_i = \int_{\Omega} (w_i u_t)_t \, d\Omega + \oint_{\partial\Omega} (w_i u_t \underline{v}) \cdot \underline{n} \, ds.$$

The first term can be expanded using the product rule to show that

$$\dot{\sigma} c_i = \int_{\Omega} w_i u_{tt} + u_t (w_i)_t \, d\Omega + \oint_{\partial\Omega} (w_i u_t \underline{v}) \cdot \underline{n} \, ds.$$

Then, by using the *Divergence Theorem*, this becomes

$$\oint_{\partial\Omega} u\underline{v} \cdot \underline{n}\, ds = \int_\Omega \underline{\nabla} \cdot (u\underline{v})\, d\Omega.$$

One can quite easily show that equation (4.7) can be further rearranged to give

$$\dot{\sigma} c_i = \int_\Omega w_i u_{tt} + u_t (w_i)_t + \underline{\nabla} \cdot (w_i u_t \underline{v})\, d\Omega.$$

The third term can then be split using the fact that $\underline{\nabla} \cdot (w_i u_t \underline{v}) = w_i u_t \underline{\nabla} \cdot \underline{v} + \underline{v}\, \underline{\nabla} \cdot w_i u_t$ to show that

$$\dot{\sigma} c_i = \int_\Omega w_i u_{tt} + u_t (w_i)_t + u_t \underline{v} \cdot \underline{\nabla} w_i + w_i \underline{\nabla} \cdot (u_t \underline{v})\, d\Omega.$$

Multiplying equation (4.6) by u_t, we can quite clearly see that two of the above terms are cancelled out and thus the equation becomes

$$\dot{\sigma} c_i = \int_\Omega w_i \left(u_{tt} + \underline{\nabla} \cdot (u_t \underline{v}) \right)\, d\Omega. \tag{4.8}$$

Now one can begin substituting equation (4.1) into the first term of (4.8), which shows that

$$\dot{\sigma} c_i = \int_\Omega w_i \left[\left(\frac{1}{a^2} \nabla^2 u + f(u) \right)_t + \underline{\nabla} \cdot (u_t \underline{v}) \right]\, d\Omega.$$

This can then be expanded to give

$$\dot{\sigma} c_i = \int_\Omega w_i \left[\frac{1}{a^2} \nabla^2 u_t + f'(u) u_t + \underline{\nabla} \cdot (u_t \underline{v}) \right]\, d\Omega.$$

A further substitution of equation (4.1) can be made, but before this a similar approach as in the one dimensional case is used to remove the complications produced by having to try to evaluate a $\nabla^4 u$ term. Once again a substitution is made, this time allowing $p = -\nabla^2 u$. The solution of p will be explained later. Substituting equation (4.1) in again produces

$$\dot{\sigma} c_i = \int_\Omega w_i \left[\frac{1}{a^2} \nabla^2 \left(\frac{1}{a^2} \nabla^2 u + f(u) \right) + f'(u) \left(\frac{1}{a^2} \nabla^2 u + f(u) \right) + \underline{\nabla} \cdot (u_t \underline{v}) \right]\, d\Omega.$$

If the substitution of $-p$ for $\nabla^2 u$ is now used the above equation becomes

$$\dot{\sigma} c_i = \int_\Omega w_i \left[\frac{1}{a^2} \nabla^2 \left(f(u) - \frac{p}{a^2} \right) + f'(u) \left(f(u) - \frac{p}{a^2} \right) + \underline{\nabla} \cdot (u_t \underline{v}) \right]\, d\Omega. \tag{4.9}$$

So far, the final term of the equation has been left untouched because it needs to be considered in a different way to the other terms. If we consider the term $\underline{\nabla} \cdot (u_t \underline{v})$, then it can be seen that, under these current conditions, $\underline{v}$ has no unique solution since the curl of an arbitrary function λ could be added to $u_t\underline{v}$ and it would not alter the solution. To ensure uniqueness, a velocity potential ψ is introduced such that

$$\underline{v} = \underline{\nabla}\psi. \tag{4.10}$$

By introducing this potential, the aforementioned last term in equation (4.9) is altered such that

$$\int_\Omega w_i \underline{\nabla} \cdot (u_t\underline{v})\, d\Omega = \int_\Omega w_i \underline{\nabla} \cdot (u_t \underline{\nabla}\psi)\, d\Omega.$$

One of Green's theorems can then be applied to this to show that

$$\int_\Omega w_i \underline{\nabla} \cdot (u_t \underline{\nabla} \cdot \psi)\, d\Omega = \oint_{\partial\Omega} w_i u_t \underline{\nabla}\psi \cdot \underline{v}\, \underline{n}\, ds - \int_\Omega u_t \underline{\nabla} w_i \cdot \underline{\nabla}\psi\, d\Omega$$

The first oterm on the right hand side is clearly zero, since u_t is zero on the boundary. This reduces equation (4.9) to

$$\dot{\sigma} c_i = \int_\Omega w_i \left[\frac{1}{a^2} \nabla^2 \left(f(u) - \frac{p}{a^2} \right) + f'(u) \left(f(u) - \frac{p}{a^2} \right) \right] d\Omega - \int_\Omega u_t \underline{\nabla} w_i \cdot \underline{\nabla}\psi\, d\Omega. \tag{4.11}$$

Next, an aspect taken from the one dimensional case must be considered. In the one dimensional case a variable Θ was introduced so that a midpoint rule could be used to calculate the new u_t value. However, since the nodes are not equally spaced, the standard midpoint rule is not particularly accurate and thus Θ was used. It is much the same case in two dimensions. Once again a variable Θ is defined, albeit it in two dimensions this time, such that

$$\Theta_i = \int_\Omega w_i u\, d\Omega. \tag{4.12}$$

Note that Ω_i represents the ringed area created between the nodes. Since this problem is being solved on a circle with a boundary condition which is uniform along the circumference it can be considered radially. Therefore, where a node previously represented a single point on a one dimensional line, each node actually represents a circle centred about the origin. These circles will start uniform distances apart, but then will obviously move once the adaptive method begins. The full geometry of this problem will be explained alongside the finite element solution.

As with one dimension, the change in Θ needs to be considered and thus equation (4.16) must be differentiated with respect to time to give

$$\dot{\Theta}_i = \frac{d}{dt}\int_\Omega w_i u \, d\Omega. \tag{4.13}$$

From the previous workings involving σ and $\dot{\sigma}$, it is clear that

$$\dot{\Theta}_i = \int_\Omega w_i \left(u_t + \underline{\nabla} \cdot (u\underline{v}) \right) \, d\Omega$$

Once again one of Green's theorems can be applied to the second term to show that

$$\dot{\Theta}_i = \int_\Omega w_i u_t - u\underline{\nabla} \cdot (w_i \cdot \underline{v}) \, d\Omega. \tag{4.14}$$

4.1 The Finite Element Part in 2 Dimensions

Currently only the test function w_i has been introduced. To complete a finite elements solution to the problem a function must actually be chosen here and in this case it will be the standard two dimensional hat function ϕ. Several equations ((4.11), (4.16) and (4.17)) need to be slightly altered to take into account this hat function. Equation (4.11) becomes

$$\dot{\sigma} c_i = \int_\Omega \phi_i \left[\frac{1}{a^2}\nabla^2 \left(f(u) - \frac{p}{a^2} \right) + f'(u)\left(f(u) - \frac{p}{a^2} \right) \right] \, d\Omega - K_i(u_t)\underline{\psi}. \tag{4.15}$$

Equation (4.12) becomes

$$\Theta_i = \int_\Omega \phi_i u \, d\Omega \tag{4.16}$$

and (by expanding ψ as $\Sigma\psi_j\phi_j$) equation (4.14) becomes

$$\dot{\Theta}_i = \int_\Omega \phi_i u_t \, d\Omega - K(u)_i \psi_i,$$

which can also (by expanding u_t as $\Sigma(u_t)_j\phi_j$) be written as

$$\underline{\dot{\Theta}} = M\underline{u}_t - K(u)\underline{\psi} \tag{4.17}$$

$K(u)$ and M will be defined later in this dissertation and $\underline{\psi}$ will be known after solving equation (4.15), thus reducing equation (4.17) to an explicit means of calculating $\dot{\Theta}_i$.

Now, if u is approximated using $u = \Sigma_{i=0}^{N}\phi_j w_j$, then equation (4.16) reduces to

$$\Theta_i = (M\underline{u})_i.$$

This further reduces to

$$\underline{\Theta} = M\underline{u}, \tag{4.18}$$

where M is the standard finite elements mass matrix.

As mentioned in an earlier section, the matrices K and M need to be defined, as well as the matrix functions $K(u_t)$ and $K(u)$. However, before the assembly of the matrices can take place, the triangulation of the region must be considered.

4.2 The Triangulation

The two dimensional problem is being considered upon a circle with radius of 1. As with the one dimensional case, the problem has been scaled and thus, regardless of the actual radius, the solution is always shown on a circle with a radius of one.

To begin the triangulation, there must be a means of choosing nodes, upon which the triangles can be based. In this case, a number (N) of 'rings' will be placed about the centre of the circle. Initially, they will all be equally separated, but once the adaptive method begins this will clearly change. Upon each of these rings, an even number (M) of nodes will be placed. Using an even number of nodes (which remains fixed) retains the symmetry of the triangulation and forces every triangle to be isosceles. These nodes will be alternately placed depending on which ring they lie upon. So for example: if four nodes are chosen, then the first ring out from the centre of the circle will have its nodes positions at 0, 90, 180 and 270 degrees. On the next ring, the nodes will be placed at 45, 135, 225 and 315 degrees. This pattern is then repeated throughout, as illustrated by figure 4.1.

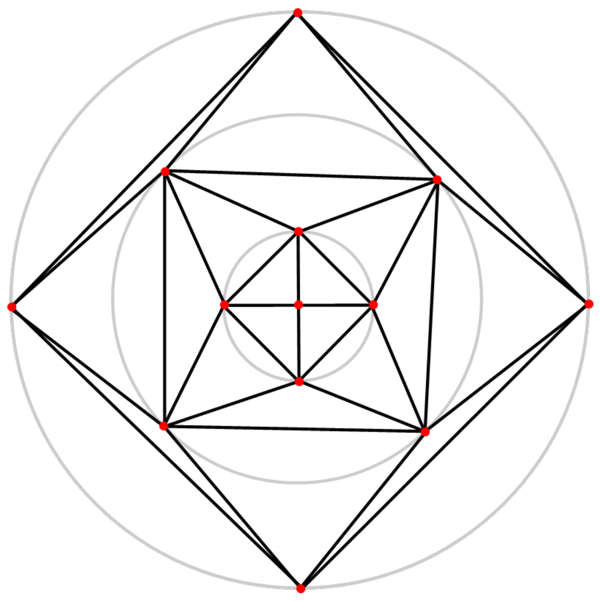
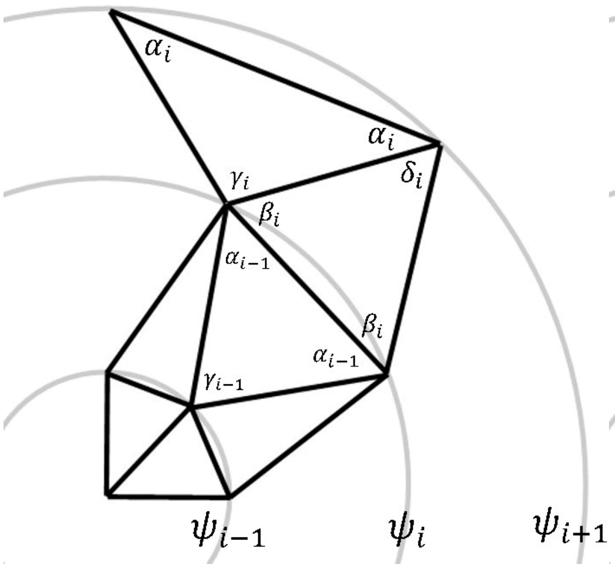
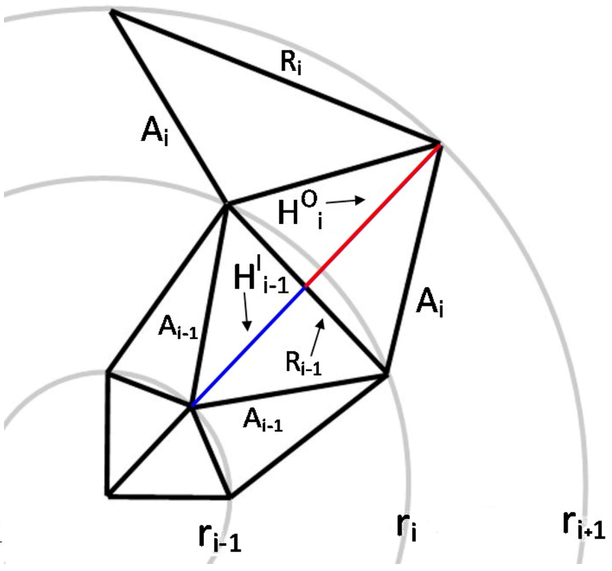


Figure 4.1: For this case, one can see that both N and M are 4. The centre point is counted as one of the rings despite not actually being a ring itself. Also note that this is a very simple example which could not be used in practice, as the angles in the larger triangles are beginning to turn obtuse, at which point the method will fail.

By considering the problem like this, all of the triangles become isosceles, thus lending a great deal of extra symmetry to the problem. However, it does mean that two different types of triangle must be considered: inward and outward pointing. Clearly, the circle created by the first ring and the centre point of the circle will only have inward pointing triangles, but from then on each area created between the rings will be made up of both inward and outward pointing triangles.



(a) The angles used in the triangulation [Note that this figure assume that it is either $K(u_t)\psi$ or $K(u)\psi$ being solved.



(b) Lengths used in the triangulation

Figure 4.2: Labelling of the triangulation

It will become clearer when assembling the aforementioned K and M matrices that every angle, side length and height for all of the triangles must be found. It was stated above that the first set of triangles (the set between the centre point and the first ring) are all inward pointing. They are also the most simple to calculate, as the lengths of the pair of equal sides in each triangle are already known, as is the single angle in each triangle.

The rest of the triangles are a little more difficult to calculate. If one considers figure 4.2, then one can see how it is possible to calculate all of the values required.

As stated above, the first set of triangles are all inward pointing and straightforward to calculate. As such, the first triangle to be considered after this is an outward pointing one. If one considers the height (H^I_{i-1}) of the inward pointing triangle which has two of its corners touching the ith ring, then one can use this to calculate the height (H^O_i) of the outward pointing triangle which shares a base with it. If one lets r_i represent the position of the ith ring, then one can see that the sum of the two heights must be equal to

$$H^I_{i-1} + H^O_i = r_{i+1} - r_{i-1}$$

At this point, the two r values are known and H_i^I can be quite easily calculated using *Pythagoras' Theorem* to show that

$$H_{i-1}^I = \sqrt{A_{i-1}^2 - \frac{R_{i-1}^2}{4}}. \tag{4.19}$$

Note that both the R and A terms are known here, since they come from previously calculated values. With H_{i-1}^I calculated, H_i^O can then be calculated using

$$H_i^O = r_{i+1} - r_{i-1} - H_{i-1}^I. \tag{4.20}$$

Since H_i^O bisects the outward pointing triangle, one can use *Pythagoras' Theorem* again to show that

$$A_i = \sqrt{\frac{R_{i-1}^2}{4} + (H_i^O)^2}. \tag{4.21}$$

Continuing with the triangle formed by the bisection, it is possible to find the remaining angles using the sine rule by showing that

$$\beta_i = sin^{-1}\left(H_i^O / A_i\right). \tag{4.22}$$

Using the fact that the triangles are all isosceles, one can then say that

$$\delta_i = \pi - 2\beta_i. \tag{4.23}$$

γ_i can now be found, using one of two conditions. If it is γ_1 being calculated, then

$$\gamma_1 = 2\pi - 2\beta_1 - 2\alpha_0, \tag{4.24}$$

otherwise

$$\gamma_i = 2\pi - 2\alpha_{i-1} - \delta_{i-1} - 2\beta_i. \tag{4.25}$$

Using γ, α can then be found using

$$\alpha_i = \frac{\pi - \gamma_i}{2}. \tag{4.26}$$

The remaining triangle base (R_i) can then be calculated using another sine rule and therefore

$$R_i = A_i \frac{sin\, \gamma_i}{sin\, \alpha_i}. \tag{4.27}$$

Finally, the loop used to calculate the values required can only begin if the lengths and angles in the original triangles are known. If there are M nodes on each ring, then

$$\gamma_0 = \frac{2\pi}{M} \tag{4.28}$$

$$\alpha_0 = \frac{\pi - \gamma_0}{2} \tag{4.29}$$

$$A_0 = r_1 \tag{4.30}$$

$$R_0 = A_0 \frac{sin\,\gamma_0}{sin\,\alpha_0}. \tag{4.31}$$

Now that the triangulation is complete, the finite elements formulation and assembly can begin.

4.3 Assembling the Matrices

As stated in an earlier section, there are several different matrices which need to be assembled. Three of these matrices (K, $K(u_t)$ and $K(u)$) are very similar and assembling just one of them gives the general form the the remaining two. M will also need to be assembled. However, we shall start by considering $K(u_t)$ first, as this will also give K and $K(u)$.

It was mentioned during the triangulation that both inward and outward pointing triangles must be considered and therefore there are two elemental matrices which need to be considered for assembly. If one starts by considering the standard finite elements K matrix in two dimensions and then multiply it by $\dot{I}_i$ (since this is $K(u_t)$) and then considers equation (4.15), then for an inward pointing triangle each element is given by

$$(K(u_t)_e^I)_i\,\underline{\psi} = \frac{\dot{I}_i}{2}\begin{pmatrix} 2\cot\alpha_i & -\cot\alpha_i & -\cot\alpha_i \\ -\cot\alpha_i & \cot\alpha_i + \cot\gamma_i & -\cot\gamma_i \\ -\cot\alpha_i & -\cot\gamma_i & \cot\alpha_i + \cot\gamma_i \end{pmatrix}\begin{pmatrix} \psi_i \\ \psi_{i+1} \\ \psi_{i+1} \end{pmatrix} = \begin{pmatrix} f_i^I \\ f_i^I \\ f_i^I \end{pmatrix}.$$

The outward pointing triangle can similarly be given by

$$(K(u_t)_e^O)_i\,\underline{\psi} = \frac{\dot{O}_i}{2}\begin{pmatrix} \cot\beta_i + \cot\delta_i & -\cot\beta_i & -\cot\delta_i \\ -\cot\beta_i & 2\cot\beta_i & -\cot\beta_i \\ -\cot\delta_i & -\cot\beta_i & \cot\delta_i + \cot\beta_i \end{pmatrix}\begin{pmatrix} \psi_i \\ \psi_{i+1} \\ \psi_i \end{pmatrix} = \begin{pmatrix} f_i^O \\ f_i^O \\ f_i^O \end{pmatrix}.$$

Where $(K(u_t)^I_e)_i$ represents the elemental matrix for the *ith* inward pointing triangle, $\dot{I}$ and $\dot{O}$ represent the midpoint values of u_t of the element. So $\dot{I}_i$ is equal to the area of the inward pointing triangle, multiplied by the average of the values of u_t on the tips of the triangle. As such

$$\dot{I}_i = \left(\frac{R_i H^I_i}{2}\right)\left(\frac{(u_t)_i + 2(u_t)_{i+1}}{3}\right) \tag{4.32}$$

and therefore $\dot{O}$ can be calculated using

$$\dot{O}_i = \left(\frac{R_{i-1} H^O_i}{2}\right)\left(\frac{2(u_t)_i + (u_t)_{i+1}}{3}\right) \tag{4.33}$$

Concentrating again on the matrices representing the elements shows that they can be simplified due to the values of ψ on every given ring having the same value. Using this symmetry reduces the two matrix systems to

$$(K(u_t)^I_e)\,_i\underline{\psi} = \dot{I}_i \begin{pmatrix} \cot\alpha_i & -\cot\alpha_i \\ -\cot\alpha_i & \cot\alpha_i \end{pmatrix} \begin{pmatrix} \psi_i \\ \psi_{i+1} \end{pmatrix} = \begin{pmatrix} f^I_i \\ 2f_i \end{pmatrix} \tag{4.34}$$

and

$$\left(K(u_t)^O_e\right)\,_i\underline{\psi} = \dot{O}_i \begin{pmatrix} \cot\beta_i & -\cot\beta_i \\ -\cot\beta_i & \cot\beta_i \end{pmatrix} \begin{pmatrix} \psi_i \\ \psi_{i+1} \end{pmatrix} = \begin{pmatrix} 2f^O_i \\ f_i \end{pmatrix}. \tag{4.35}$$

Now, to consider each area (between two rings) of triangles as a single matrix, (4.34) and (4.35) must be combined. This results in

$$(K(u_t)_e)_i\;\underline{\psi} = \begin{pmatrix} g_i & -g_i \\ -g_i & g_i \end{pmatrix} \begin{pmatrix} \psi_i \\ \psi_{i+1} \end{pmatrix} = \begin{pmatrix} f^I_i + 2f^O_i \\ 2f^I_i + f^O_i \end{pmatrix}, \tag{4.36}$$

where

$$g_i = \dot{I}_i \cot\alpha_i + \dot{O}_i \cot\beta_i \tag{4.37}$$

for $i = 1$ to N.

Now that the matrix system is known at each element it is possible to assemble the elements to form a system covering the entire domain. However, one must first note that the first element is different from the rest, as only inward pointing triangles exist between the centre of the circle and the first ring. Therefore the assembly begins with

$$\begin{pmatrix} \dot{I}_0\cot\alpha_0 & -\dot{I}_0\cot\alpha_0 \\ -\dot{I}_0\cot\alpha_0 & \dot{I}_0\cot\alpha_0 \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \end{pmatrix} = \begin{pmatrix} f^I_0 \\ 2f^I_0 \end{pmatrix}.$$

Adding the next element to this produces

$$\begin{pmatrix} \dot{I}_0 \cot\alpha_0 & -\dot{I}_0 \cot\alpha_0 & 0 \\ -\dot{I}_0 \cot\alpha_0 & \dot{I}_0 \cot\alpha_0 + g_1 & -g_1 \\ 0 & -g_1 & g_1 \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \end{pmatrix} = \begin{pmatrix} f_0^I \\ 2f_0^I + f_1^I + 2f_1^O \\ 2f_1^I + f_1^O \end{pmatrix}.$$

Adding a second element shows

$$\begin{pmatrix} \dot{I}_0 \cot\alpha_0 & -\dot{I}_0 \cot\alpha_0 & 0 & 0 \\ -\dot{I}_0 \cot\alpha_0 & \dot{I}_0 \cot\alpha_0 + g_1 & -g_1 & 0 \\ 0 & -g_1 & g_1 + g_2 & -g_2 \\ 0 & 0 & -g_2 & g_2 \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \psi_3 \end{pmatrix} = \begin{pmatrix} f_0^I \\ 2f_0^I + f_1^I + 2f_1^O \\ 2f_1^I + f_1^O + f_2^I + 2f_2^0 \\ 2f_2^I + f_2^O \end{pmatrix}.$$

Adding this second full element gives the general form of the system and therefore one can show that

$$K(u_t)\underline{\psi} = \begin{pmatrix} \dot{I}_0 \cot\alpha_0 & -\dot{I}_0 \cot\alpha_0 & 0 & \cdots & & & & 0 \\ -\dot{I}_0 \cot\alpha_0 & \dot{I}_0 \cot\alpha_0 + g_1 & -g_1 & 0 & \cdots & & & 0 \\ 0 & -g_1 & g_1 + g_2 & -g_2 & 0 & \cdots & & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & \cdots & & 0 \\ 0 & 0 & -g_{i-1} & g_{i-1} + g_i & -g_i & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & & -g_{N-1} \\ 0 & & & & & & -g_{N-1} & g_{N-1} \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \vdots \\ \psi_i \\ \vdots \\ \psi_N \end{pmatrix} = \underline{f} \tag{4.38}$$

where

$$\underline{f} = \begin{pmatrix} f_0^I \\ 2f_0^I + f_1^I + 2f_1^0 \\ 2f_1^I + f_1^0 + f_2^I + 2f_2^O \\ \vdots \\ 2f_{i-1}^I + f_{i-1}^0 + f_i^I + 2f_i^O \\ \vdots \\ 2f_{N-1}^I + f_{N-1}^O \end{pmatrix}. \tag{4.39}$$

Note that the subscripts in the K and $\underline{f}$ matrices both have maximum values of $N-1$. This is due to the rings in the domain being labelled from 0 to N, with the elements in the first area being labelled as element 0. Clearly, if the final ring is labelled N, then the final elements are actually labelled as $N-1$, as that is the $(N-1)$th area in the domain.

To begin solving $K(u_t)\underline{\psi} = \underline{f}$, both f_i^I and f_i^O must be defined. Both terms come from equation (4.15), but differ due to the ϕ_i term. If one rearranges equation (4.15) so that $K(u_t)_i\psi_i$ is the subject then one has

$$K_i(u_t)\underline{\psi} = \int_\Omega w_i \left[\frac{1}{a^2}\nabla^2\left(f(u) - \frac{p}{a^2}\right) + f'(u)\left(f(u) - \frac{p}{a^2}\right)\right] d\Omega - \dot{\sigma} c_i. \tag{4.40}$$

The $\dot{\sigma}$ term is simple enough for each triangle, but the integral is a little more complicated. If the integral has u_t reintroduced and then the two terms split, then

$$\int_\Omega \phi_i \left[\frac{1}{a^2}\nabla^2\left(f(u) - \frac{p}{a^2}\right) + f'(u)\left(f(u) - \frac{p}{a^2}\right)\right] d\Omega = \frac{1}{a^2}\int_\Omega \phi_i \nabla^2 u_t \, d\Omega + \int_\Omega \phi_i f'(u) u_t \, d\Omega.$$

Expanded like this, the second integral is now also simple to calculate for any given triangle, since it is simply one third of the area of the triangle, multiplied by $f'(u)u_t$ on the ith ring. The first term can be solved in a similar way, but using the symmetry of the problem to approximate the $\nabla^2 u_t$ term. Currently, the problem is being considered on a Cartesian domain, but the problem is known to be radially symmetric and is being solved along the radius. Therefore, rather than using $\nabla^2 u_t$,

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u_t}{\partial r}\right)$$

will be considered instead. The derivation has already been seen for this in the preliminary results section. Therefore the f values in the system can be given by

$$f_i^I = F_i^I \left(\frac{2}{r_j a^2 (r_{i+1} - r_{i-1})}\left[r_{i+1/2}\left(\frac{(u_t)_{i+1} - (u_t)_i}{r_{i+1} - r_i}\right) - r_{i-1/2}\left(\frac{(u_t)_i - (u_t)_{i-1}}{r_i - r_{i-1}}\right)\right] + f'(u)(u_t)_i\right) + \dot{\sigma} c_i, \tag{4.41}$$

where

$$F_i^I = \frac{1}{6} H_i^I R_i \tag{4.42}$$

and $r_{i+1/2} = \frac{1}{2}(r_{j+1} + r_j)$. Similarly

$$f_i^O = F_i^O \left(\frac{2}{r_j a^2 (r_{i+1} - r_{i-1})}\left[r_{i+1/2}\left(\frac{(u_t)_{i+1} - (u_t)_i}{r_{i+1} - r_i}\right) - r_{i-1/2}\left(\frac{(u_t)_i - (u_t)_{i-1}}{r_i - r_{i-1}}\right)\right] + f'(u)(u_t)_i\right) + \dot{\sigma} c_i, \tag{4.43}$$

where

$$F_i^O = \frac{1}{6} H_i^O R_{i-1}. \tag{4.44}$$

Now that $K(u_t)$ is known, $K(u)$ from equation (4.17) can also be stated.

$K(u)$ given by

$$K(u)\underline{\psi} = \begin{pmatrix} I_0\cot\alpha_0 & -I_0\cot\alpha_0 & 0 & \cdots & & & & 0 \\ -I_0\cot\alpha_0 & I_0\cot\alpha_0 + h_1 & -h_1 & 0 & \cdots & & & 0 \\ 0 & -h_1 & h_1+h_2 & -h_2 & 0 & \cdots & & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & \cdots & & 0 \\ 0 & 0 & -h_{i-1} & h_{i-1}+h_i & -h_i & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & & -h_{N-1} \\ 0 & & & & & & -h_{N-1} & h_{N-1} \end{pmatrix}, \tag{4.45}$$

where

$$h_i = I_i \cot\alpha_i + O_i \cot\beta_i, \tag{4.46}$$

$$I_i = \left(\frac{R_i H_i^I}{2}\right)\left(\frac{u_i + 2u_{i+1}}{3}\right) \tag{4.47}$$

and

$$O_i = \left(\frac{R_{i-1} H_i^O}{2}\right)\left(\frac{2u_i + u_{i+1}}{3}\right). \tag{4.48}$$

4.4 Considering p

It was stated earlier that $p = -\nabla^2 u$. As with the one dimensional case (in which $p = v$), p will be solved using a finite element method. So far one has

$$p = -\nabla^2 u. \tag{4.49}$$

If this is multiplied through by a test function w_i and then integrated one has

$$\int_\Omega w_i p \, d\Omega = -\int_\Omega w_i \nabla^2 u \, d\Omega.$$

Using one of Green's theorems this can be rewritten as

$$\int_\Omega w_i p \, d\Omega = \int_\Omega \underline{\nabla} w_i \cdot \underline{\nabla} u \, d\Omega - \oint_{d\Omega} w_i \underline{\nabla} u \, ds.$$

If the two dimensional hat function is introduced, this becomes

$$\int_\Omega \phi_i p \, d\Omega = \int_\Omega \underline{\nabla} \phi_i \cdot \underline{\nabla} u \, d\Omega - \oint_{d\Omega} \phi_i \underline{\nabla} u \cdot \underline{ds}.$$

Since the hat function is not 0 around the boundary, the second term on the right hand side is removed by weakly imposing $\underline{\nabla} u = 0$ on the boundary, thus reducing the problem to

$$\int_\Omega \phi_i p \, d\Omega = \int_\Omega \underline{\nabla}\phi_i \cdot \underline{\nabla} u \, d\Omega \tag{4.50}$$

This reduces to the matrix form

$$M\underline{p} = K\underline{u},$$

where K is the standard finite elements stiffness matrix and M is a standard mass matrix. Therefore K is given by

$$K = \begin{pmatrix} \cot\alpha_0 & -\cot\alpha_0 & 0 & \cdots & & & & 0 \\ -\cot\alpha_0 & \cot\alpha_0 + C_1 & -C_1 & 0 & \cdots & & & 0 \\ 0 & -C_1 & C_1 + C_2 & -C_2 & 0 & \cdots & & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & \cdots & & 0 \\ 0 & 0 & -C_{i-1} & C_{i-1} + C_i & -C_i & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & & -C_{N-1} \\ 0 & & & & & & -C_{N-1} & C_{N-1} \end{pmatrix}, \tag{4.51}$$

where

$$C_i = \cot\alpha_i + \cot\beta_i \tag{4.52}$$

4.5 Considering The Mass Matrix

So far three different K matrices have been considered and assembled, but to complete the finite elements solution the M matrix must be assembled, as this is required for the solutions of Θ, p and subsequently u. To assemble M, we shall consider it using the equation (4.16). From equation (4.16), we know that

$$\Theta_i = \int_\Omega \phi_i u \, d\Omega$$

and therefore

$$\underline{\Theta} = M\underline{u}.$$

This is a standard finite element result, however the matrix produced does need to be assembled, since there are multiple different sizes of triangles to be considered. Generally an elemental matrix is given by

$$M = \frac{triangle\,area}{12}\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}. \tag{4.53}$$

However, there are two different types of triangle and so a different area is considered for each one. F_i^I represents the area of the *ith* inward pointing triangle and F_i^O the *ith* outward. This produces two elemental matrices, such that

$$(M_i^I)_e u = \frac{F_i^I}{12}\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}\begin{pmatrix} u_i \\ u_{i+1} \\ u_{i+1} \end{pmatrix},$$

which simplifies to

$$(M_i^I)_e u = \frac{F_i^I}{6}\begin{pmatrix} 1 & 1 \\ 1 & 3 \end{pmatrix}\begin{pmatrix} u_i \\ u_{i+1} \end{pmatrix}. \tag{4.54}$$

Similarly the inward element is given by

$$(M_i^O)_e u = \frac{F_i^O}{12}\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}\begin{pmatrix} u_i \\ u_{i+1} \\ u_i \end{pmatrix},$$

which simplifies to

$$(M_i^O)_e u = \frac{F_i^O}{6}\begin{pmatrix} 3 & 1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} u_i \\ u_{i+1} \end{pmatrix}. \tag{4.55}$$

Combining these two results in the full elemental matrix for M, which can now be shown to be

$$M_e u = \frac{1}{6}\begin{pmatrix} F_i^I + 3F_i^O & F_i^I + F_i^O \\ F_i^I + F_i^O & 3F_i^I + F_i^O \end{pmatrix}\begin{pmatrix} u_i \\ u_{i+1} \end{pmatrix}. \tag{4.56}$$

If one lets $F_i = F_i^I + F_i^O$, then as with the K matrices, the first element only contains inward pointing triangles and therfore

$$M_0 u = \frac{1}{6}\begin{pmatrix} F_0^I & F_0^I \\ F_0^I & 3F_0^I \end{pmatrix}\begin{pmatrix} u_0 \\ u_1 \end{pmatrix}.$$

Adding the next full element to this gives

$$Mu = \frac{1}{6}\begin{pmatrix} F_0^I & F_0^I & 0 \\ F_0^I & 3F_0^I + F_i^I + 3F_1^O & F_1 \\ 0 & F_1 & 3F_1^I + F_1^O \end{pmatrix}\begin{pmatrix} u_0 \\ u_1 \\ u_2 \end{pmatrix}.$$

Adding a further element produces

$$Mu = \frac{1}{6}\begin{pmatrix} F_0^I & F_0^I & 0 & 0 \\ F_0^I & 3F_0^I + F_i^I + 3F_1^O & F_1 & 0 \\ 0 & F_1 & 3F_1^I + F_1^O + F_2^I + 3F_2^O & F_2 \\ 0 & 0 & F_2 & 3F_2^I + F_2^O \end{pmatrix}\begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{pmatrix}.$$

From this, one can see the general form of the matrix is

$$M = \frac{1}{6}\left(\begin{array}{cccccc} F_0^I & F_0^I & 0 & \cdots & & \\ F_0^I & 3F_0^I + F_1^I + 3F_1^O & F_1 & 0 & \cdots & \\ 0 & F_1 & 3F_1^I + F_1^O + F_2^I + 3F_2^O & F_2 & 0 & \cdots \\ & \ddots & \ddots & \ddots & & \\ & 0 & F_{i-1} & 3F_{i-1}^I + F_{i-1}^O + F_i^I + 3F_i^O & F_i & 0 \\ & & \ddots & \ddots & \ddots & \\ & & & & F_{N-1} & 3F_{N-1}^I + \end{array}\right. \tag{4.57}$$

From this, Θ can be solved using a tri-diagonal solver applied to the system

$$\underline{\Theta} = M\underline{u},$$

where

$$u = \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_i \\ \vdots \\ u_N \end{pmatrix}$$

4.6 The Two Dimensional Solution

Now that all of the required variables have been defined, a method for the solution of the problem can begin. As with the one dimensional case, a twice differentiable function has been chosen and thus the u values can be immediately set. However, the u_t values are not so simple to set initially. In the one dimensional case the sine function being used could simply be differentiated twice with respect to x. The initial function for u being used here is not so easy to differentiate, as it contains both an x and a y term. Fortunately, the symmetry of this problem allows it to be considered radially and thus the initial condition can also be stated as $\frac{1}{10}\cos\left(\frac{\pi}{2}r\right)$.

It was stated in the preliminary section of this paper that the $\nabla^2 u$ term can be replaced with with

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right). \tag{4.58}$$

Substituting in the radial initial condition for u gives

$$\frac{1}{r}\frac{\partial}{\partial r}\left(-\frac{r\pi}{20}\sin\left(\frac{\pi}{2}r\right)\right) \tag{4.59}$$

Applying the product rule to this produces.

$$-\frac{1}{r}\left(\frac{\pi}{20}\sin\left(\frac{\pi}{2}r\right)+\frac{r\pi^2}{40}\sin\left(\frac{\pi}{2}r\right)\right). \tag{4.60}$$

Once in this form, it can then be seen that the initial u_t can be written as

$$u_t = f(u) - \frac{1}{a^2}\left(\frac{\pi}{r20}\sin\left(\frac{\pi}{2}r\right)+\frac{\pi^2}{40}\sin\left(\frac{\pi}{2}r\right)\right). \tag{4.61}$$

Once the initial conditions are set, both the initial σ, Θ and c_i values can be calculated.

The boundary conditions however, are slightly different. For the one dimensional case both $x = 0$ and $x = 1$ had the condition $u = 0$ imposed upon them and therefore u_t was also equal to zero at the boundary. In this, two dimensional case, only the condition at the outer boundary of the circle ($r = 1$) will have the condition $u = 0$ on it. This obviously leads to $u_t = 0$ on this boundary as well.

Once the initial condition is set, the first p values are already known since $p = -\nabla^2 u$ and this has already been calculated using a radial form when finding the initial u_t. Therefore the initial p values are given by $\frac{\pi}{r20}\sin\left(\frac{\pi}{2}r\right)+\frac{\pi^2}{40}\sin\left(\frac{\pi}{2}r\right)$. This will give the p

estimate in u_t. Once this is known, $\dot{\sigma}$ can be calculated. Equation (4.15) only refers to each area between the rings on the circle. If equation (4.15) is summed from 0 to N, then the $K(u_t)$ term is removed since all the rows sum up to 0. Similarly, the ϕ_i terms all sum to 1, thus reducing the equation to

$$\dot{\sigma} c = \int_\Omega \left[\frac{1}{a^2} \nabla^2 \left(f(u) - \frac{p}{a^2} \right) + f'(u) \left(f(u) - \frac{p}{a^2} \right) \right] d\Omega, \tag{4.62}$$

where c is simply the sum of all the c_i values.

Once $\dot{\sigma}$ is known, equation (4.40) can be solved and therefore the system $K(u_t)\underline{\psi} = \underline{f}$ can be solved using a tri-diagonal solver.

This solution will find ψ and thus the solution of $\dot{\Theta}$ can begin found by considering equation (4.17). It has already been stated that this solution is explicit, since all of the right hand terms are known.

Once ψ is known, v can be found since $\underline{v} = \underline{\nabla} \cdot \underline{\psi}$. Once again the symmetry of the problem can be employed to simplify this calculation. Obviously $\underline{\nabla} \cdot \underline{\psi}$ is made up of the two components $\frac{\partial \psi}{\partial x}$ and $\frac{\partial \psi}{\partial y}$, but exploiting the symmetry means that one of these terms is 0. If for example, we assume the radius we are considering is in fact the x axis, then $\frac{\partial \psi}{\partial y} = 0$ and $\frac{\partial \psi}{\partial x} = \frac{\partial \psi}{\partial r}$. Therefore v can be found by simply considering a central finite difference scheme on ψ.

It should also be noted when calculating ψ that K is singular and therefore cannot be inverted as it is. It is known that the outer ring and the origin will not shift and thus $\underline{\nabla}\psi = 0$ at these points. However, this tells us little about the actual value of ψ at these points. As such ψ_N is assumed to be zero so that the K matrix can be inverted.

With all of the above calculated, σ, $\underline{r}$ and $\underline{\dot{\Theta}}$ can all be updated. All three are updated using a first order Eulerian method. As such,

$$\sigma^{n+1} = \sigma^n + \Delta t \, \dot{\sigma}^n, \tag{4.63}$$

$$\underline{r}^{n+1} = \underline{r}^n + \Delta t \, \underline{v}^n \tag{4.64}$$

and

$$\underline{\Theta}^{n+1} = \underline{\Theta}^n + \Delta t \, \underline{\dot{\Theta}}^n. \tag{4.65}$$

Now $\underline{u}$ can be calculated. Unlike the one dimensional method, this does not need to use any form of midpoint rule, as the finite elements formulation produced equation (4.18), which suggests that

$$M\underline{u} = \underline{\Theta}. \tag{4.66}$$

$\underline{\Theta}$ has already been updated and therefore this can be solved using a tri-diagonal solver, with the final row and column of the M matrix ignored, since u is known to be 0 on the boundary.

With this time step effectively over u_t can be updated. In some ways the solution for this more elegant than that of the one dimensional case. If one considers equation (4.7) and adds in the hat function ϕ then one has

$$\sigma c_i = \int_\Omega \phi_i u_t \, d\Omega.$$

When summed from $i = 0$ to N, the integral becomes the standard finite elements mass matrix and therefore

$$\sigma \underline{c} = M\underline{u}_t.$$

Clearly this is another system which can be solved using a tri-diagonal solver. However, the system is slightly different from the M given in an earlier section since u_t is known to be 0 on the boundary and therefore the last row and column of the M matrix can be ignored.

Following u_t is the recalculation of p (and subsequently the estimate for u_t using p) and then the re-triangulation of the domain using the new positions of the rings.

As with the one dimensional case, the pseudo code for this method is in Appendix 2.

4.7 Results

As with the one dimensional case the method was coded up using C++ and like the one dimensional case, the method produced nothing of use. However, whereas the one dimensional case has almost certainly been calculated correctly, the failure of the two dimensional case is less clear. Within one time step the mesh points have moved far too far (well outside the domain in fact), which (assuming the programming is working for the most part) is due to the calculation of $\dot{\sigma}$. More about this will be said in the critical analysis section of this method.

4.8 Critical Analysis

In the case of the two dimensional method it does appear that it is the program at fault, or at least this is the easiest assumption to make. When running the program, the $\dot{\sigma}$ value becomes very large and negative. Further exploration into this revealed that it is not the coding causing this outright. $\dot{\sigma}$ is calculated by applying the trapezium rule across the radius and then using the radial symmetry of the problem to generate the volume under the curve. So for example, the method starts by calculating the area under the curve between r_0 and r_1. This value is then multiplied by πr_1^2. Therefore the volume of the region between the origin and the first ring is given. The second ring is then calculated by considering the area under the curve (along the radius) between r_0 and r_2 and then multiplying this by πr_2^2. The value from the previous area is then subtracted from this to give the volume under the curve on the annulus between r_1 and r_2. This is then repeated across the domain.

However, problems begin to arise when arriving at the penultimate area, as this requires the value of u_t on the boundary. u_t is zero on the boundary and this forces the radial approximation to $\nabla^2 p$ to generate a rather large, negative value. There is also the added issue of calculating the value of $\nabla^2 p$ on the boundary.

As such, it would appear that attempting to create a solution like this is not particularly viable. The integral used to calculate $\dot{\sigma}$ would need a different method of solution, which includes some method to approximate the $\nabla^2 u_t$ term on the boundary.

Other than this, it is difficult to speculate on this method, as the results produced are so poor.

Chapter 5

Conclusions and Futher Work

5.1 Conclusions

This dissertation has covered the use of an r-adaptive method for solving a particular quenching problem for both one and two dimensions. Initial tests and papers by [7, 9] gave an indication as to where in the domain the problem would quench and at what point in time the quenching point should be reached.

Both one and two dimensional methods were attempted. The two dimensional method added little to the study other than to illustrate that the boundary conditions play a huge part in trying to calculate a solution if u_t is taken as the monitor function.

The one dimensional method on the other hand, did provide some results and thus several conclusions can be drawn. Though the results produced were relatively poor and the method required some alterations to function, it does imply that u_t could be used as a monitor function, but not successfully. Assuming the coding used was indeed correct, then the method is hugely unstable and prone to mesh tangling. Using smaller time steps and a greater number of nodes does little to alleviate this. In fact, in the case of adding extra nodes, once over around 21 nodes the method becomes totally unusable.

Assuming the code is operating correctly, the one dimensional method could be seen to indicate that u_t should not be used as a monitor function as it appears to create a hugely unstable mesh. There is a fairly simple explanation for this. u_t is made up of both a u_{xx}/a^2 term and $f(u)$. $f(u)$ is known to be monotonically increasing from [9] and is always largest at the centre of the domain. As such, this is the term contributing a great deal to the blow up, as while it gets larger itself, it also forces u_{xx} to grow larger at the centre of the domain.

However, the forcing of u and u_t to be zero at the boundaries also creates two other large values of u_{xx}. In terms of the solution, this is not actually an issue because if u is forced to zero at the boundaries, then u_{xx} and thus u_t will be reasonably large at the first few internal points. However, in terms of the moving mesh, this is an issue, because the method is attempting to cluster the points about three areas. Although it is not the u_t itself causing this issue, but the u_{tt} values which are required by the method. The gradient of u_t is very large near the boundaries, because of the forcing of u and u_t to zero, thus forcing the u_{xx} and $\dot{\sigma}$ values to become large and therefore forces the method to fail.

As such, the main conclusion which can be drawn from this study is that using u_t as a monitor function is incredibly dependent on the boundary conditions of the problem. Using $u = 0$ at the boundaries creates such large changes in u_t that the method soon becomes unstable. What this study does allow though, is to consider further work in this area.

5.2 Further Work

Having completed this study, the list of possible alterations and future work is vast. With hindsight it is clear that aspects of this study should have been approached differently. The poor results produced by the one dimensional adaptive method are difficult to explain, since it it difficult to know whether it is the scheme itself which is poor, the programming or whether the scheme simply does not suit the equation it was designed to solve. Since it is difficult to conclude where the issue is within the method, it would seem sensible to derive the method once more, but begin with a generic PDE of the form $u_t = Lu$. Once derived like this, a PDE with a known analytic solution could be chosen as a test for the method. Possibly the largest shortcoming of this study, especially in the case of the one dimensional adaptive method, was to not approach the method in such a way that it could solve a generic PDE, rather than the specific case taken from Kawarada and Liang et al.

The one dimensional case was also peculiar, in that it used an array of different methods for solving different parts of the scheme. It would be interesting to see whether this could be streamlined to perhaps use just finite differences for the most part, with the trapezium rule aspect still required to solve the numerical integrations. The bulk of the movement of the mesh would remain the same (although v would be approximated using finite differences rather than finite elements), but a finite difference scheme on an uneven grid could certainly be considered in one dimension and could perhaps prove more accurate than the midpoint rule used.

The method used here is that of a mesh movement or motion scheme. A further study could be to redefine the scheme entirely by continuing with u_t as a monitor function, but from there on the method would differ as the aim would be to refine the localities with the greatest rate of change. It is clear from the literature that using an arc length monitor function has provided some success with regards to refinement methods, so it should prove interesting to attempt to refine a mesh using a new monitor function.

Adaptive time stepping has not been used in this case, but it is another aspect which could improve this scheme. The u_t values growing too rapidly at certain points has been seen to force the mesh to overlap. Using an adaptive time stepping scheme, even a simple method, could improve the way the mesh moves. A scheme considered during this study would have centred around applying a universal change in time step should the change under u_t become too great. This would be crude, but being so simple it would only take a small amount of work to implement it. Equally, it would be possible to apply a much more complex method, similar to that of [9] to the adaptive time stepping.

Also, using u_t as monitor contains two terms causing growth. u_{xx} is large nearest the boundaries, due to the drop off to 0, but $f(u)$ is largest in the centre of the domain. This suggests that u_t is quite large at three different points, which could be forcing the mesh to shift poorly. If the problem being solved is one of the form of (1.1), perhaps using $f(u)$ as the monitor function would prove more effective.

It was also noted during this dissertation that the use of two different methods for calculating u_t is somewhat redundant. However, this may not have made sense here, but if $f(u)$ is the monitor function then the method starts to become more useful. In this case, (where u_t is the monitor), u_t was found and then a forward Euler method was used to update u. However, $f(u)$ is a function of u and therefore the midpoint method used in the one dimensional method could actually be used to find u, rather than the Eulerian method.

Bibliography

[1] M. J. Baines, M. E. Hubbard and P. K. Jimack, *A Moving Mesh Finite Element Algorithm for the Adaptive Solution of Time-Dependent Partial Differential Equations with Moving Boundaries*, Applied Numerical Mathematics (2005), 54 (3-4). pp. 450-469.

[2] C. J. Budd, W. Huang and R. D. Russell, *Adaptivity with moving grids*, Acta Numerica (2009), pp. 1-131.

[3] W. Cao, W. Huang and R. D. Russell, *A Moving Mesh Method Based On The Geometric Conservation Law*, SIAM J. Sci. Comput., Vol. 24, No. 1 pp. 118-142.

[4] C. Y. Chan, *New results in quenching*, Proc. 1st World Congress Nonlinear Anal., de Gruyeter, Berlin, 1996, 427-434.

[5] C. Y. Chan and Lan Ke, *Beyond quenching for singular reaction-diffusion problem*, Mathematical Methods in the Applied Sciences, 17(1994), 1-9.

[6] S. L. Cole, *Blow-up in a Chemotaxis Model Using a Moving Mesh Method*, Reading University, Dissertation, 2009.

[7] K. Deng and H. Levine, *On The Blowup of u_t at Quenching*, The American Mathematical Society, Volume 106, No. 4 (August 1989), pp. 1049-1056.

[8] H. Kawarada, *On the solutions of initial-boundary value problems for $u_t = u_x x + \frac{1}{1-u}$*, Res. Inst. Math. Sci 10 (1975), 729-736.

[9] K. W. Liang, P. Lin and R. C. E Tan, *Numerical Solution of Quenching Problems Using Mesh-Dependent Variable Temporal Steps*, Applied Numerical Mathematics, Volume 57, Issues 5-7, May 2007, Pages 791-800.

[10] G.D. Smith, *Numerical Solution of Partial Differential Equations: Finite Difference Methods*, Third Edition, Oxford University Press, ISBN 0-19-859650-2

Chapter 6

Appendices

6.1 Appendix 1

6.1.1 Pseudo Code for the One Dimensional Adaptive Method

```
• Parameters are set ($N$, $\Delta x$, $\Delta t$).
• The $x$ positions are set.
• Initial $u$ values ($u = \frac{1}{10} sin\,(\pi x)$) are set.
• Initial $v$ values ($v = \frac{\pi^2}{10} sin\,(\pi x)$) are set.
• Initial $u_t$ values ($u_t = f(u) - \frac{v}{a^2}$) are set.
• $A_i$ values are calculated using a combination of the exact integral of $u_{xx}$ and the
  trapezium rule.
• $\sigma$ is calculated by summing the $A_i$ values.
• $C_i$ values are calculated by dividing each $A_i$ value by $\sigma$.
• The initial $\Theta$ values are calculated.
• The time step is moved forward to $\Delta t$ and the time loop begins.
    – $\dot{\sigma}$ is calculated.
    – $\sigma$ is updated.
```

- The $\dot{x}$ values are calculated.
- The x positions are updated.
- The $\dot{\Theta}$ values are calculated.
- The Θ values are updated.
- The u values are updated.
- The u_t values are calculated using the midpoint method.
- The v values are calculated.
- The v estimate of u_t is calculated using $vu_t = f(u) - \frac{v}{a^2}$.
- The time step is advanced and the loop begins again.

6.2 Appendix 2

6.2.1 Pseudo Code for the Two Dimensional Adaptive Method

- Parameters are set (N, Δx, Δt).
- The r positions are set.
- The initial u values ($u = \frac{1}{10}\cos\left(\frac{\pi}{2}r\right)$ are set.
- The initial p values are set by approximating $\nabla^2 u$ using a radial version of the derivative.
- The initial u_t values are calculated using $u_t = f(u) - \frac{p}{a^2}$.
- The initial c values are calculated.
- The triangulation of the domain takes place.
- The initial Θ values are calculated.
- The time loop begins.
  - $\dot{\sigma}$ is calculated.
  - The ψ values and subsequently the v values are calculated.
  - σ and the r positions are updated.
  - The $\dot{\Theta}$ values are calculated.
  - The Θ values are updated.
  - The u values are updated.
  - The p values are updated.
  - The approximation of u_t using p ($pu_t = f(u) - \frac{p}{a^2}$) is calculated.
  - The u_t values are updated using the tri-diagonal solver.
  - The time step is advanced and the loop begins again.