University of Reading

School of Mathematics, Meteorology and Physics

NUMERICAL SCHEMES FOR A NON-LINEAR DIFFUSION PROBLEM

by

Kylie Osman

August 2005

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

Declaration

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

Kylie Osman

Acknowledgements

This project was supervised by Professor Mike Baines whom I would like to thank for the interesting supervisions and his help throughout the course. Thanks also go to Wayne Gaudin from AWE for his continued help, especially with this project; his straightforward explanations were greatly appreciated. Finally I would like to thank Sue Davis and the MSc students from the past two years for keeping me sane during this course.

Abstract

Nonlinear diffusion has many different applications. When the diffusion coefficient is of the form u^n then we can find a self-similar solution which is an attractor for all general solutions. In general, numerical schemes must be used to find the solution. In this dissertation, two different types of schemes are considered; the first being Crank-Nicolson with different nonlinear solvers and the second being a moving mesh method. In the Crank-Nicolson case, I will look at two formulations, one that is conservative and one that is non-conservative. Most of the schemes used are found to work well and the solutions are found to tend to the self-similar solution in a satisfactory way.

Contents

List of Figures

1 Introduction

1.1 An Introduction to Non-Linear Diffusion

Non-linear diffusion is characterised by the partial differential equation

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(D(u)\frac{\partial u}{\partial x}\right). \tag{1.1}$$

where $D(u)$ is the diffusion coefficient. When $D(u) = u^n$, this equation is also known as the Porous Medium Equation (PME),

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(u^n\frac{\partial u}{\partial x}\right). \tag{1.2}$$

1.1.1 Gas Diffusion through a Porous Medium

The PME can be derived by considering the diffusion of a gas through a porous medium under the action of Darcy's law relating the velocity to the pressure gradient. The flow of gas is characterised in terms of pressure, p; density, u and velocity, v. The gas can be assumed to obey the conservation of mass equation

$$\rho\frac{\partial u}{\partial t} + \frac{\partial}{\partial t}(uv) = 0, \tag{1.3}$$

where ρ is the constant porosity of the medium. The gas also obeys Darcy's law

$$\mu v = -\kappa\frac{\partial p}{\partial t} \tag{1.4}$$

which is an empirical law for the dynamics of the flow through a porous medium. Here, μ is the viscosity of the gas and κ is the permeability of the medium; both these are assumed to be constant. The gas is assumed to be ideal so

$$p = p_0 u^{\lambda}, \tag{1.5}$$

where p_0 is the reference pressure and λ is the ratio of the specific heats for the gas. Substituting equations (1.4) and (1.5) into (1.3) gives

$$\frac{\partial u}{\partial t} = c\frac{\partial}{\partial t}\left(u^{\lambda}\frac{\partial u}{\partial t}\right)$$

where

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

The constant c can be scaled out of the problem and taking u^m for $D(u)$ we have equation (1.1) [12].

1.1.2 Insect Dispersal

Diffusion models also form a reasonable basis for studying insect dispersal. One use of the diffusion model of particular relevance to insects is when there is an increase in diffusion due to population pressure. For example, the diffusion coefficient $D(u)$ depends on the population density u in such a way that D increases as u increases, a typical form for $D(u)$ being u^m where $m > 0$. The dispersal patterns for grasshoppers behave similarly to this model [3].

1.1.3 Radiation Diffusion

We are particularly interested in solving the transport of radiation through various environments. The full transport equation describes the motion of radiation in detail but is expensive to solve numerically, whether by Monte Carlo methods or by some deterministic method such as Sn or Pn.

In cases when the mean free path of a photon is much smaller than a computational cell then an approximation can be made to the transport equation which yields the diffusion equation. In this approximation causality is no longer significant although in low opaque environments a radiation wave can exceed the speed of light and care must be taken when the approximation is applied. However, in a genuinely diffusive regime, it provides an adequate description of the radiation transport.

In the diffusion approximation, the diffusion coefficient is not a simple constant or linear function, but depends on the type of material and the density and temperature of the media being traversed. The dependence on temperature is roughly proportional to the 4th power and so the diffusion equation is highly non-linear. Hence to solve the diffusion equation with linear solvers the timestep must either be small so that the system is effectively linearised or an iterative scheme devised that takes into account the non-linear nature of the coefficient [10].

In practice, the temperature dependence is not evaluated via a polynomial function, but taken from tables derived from theory and experiment. Intermediate values are interpolated. In this case the derivative of the coefficient must be evaluated numerically if it is involved in the solving iteration.

I shall first be considering the case when the diffusion coefficient is known analytically to be u^4. The partial differential equation becomes

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(u^4 \frac{\partial u}{\partial x}\right). \tag{1.6}$$

I shall later look at using a table to evaluate $D(u)$.

1.2 Overview of the Report

In chapter two, I shall give the derivation of the self-similar solution for equation (1.6). This gives a family of analytic solutions which can be used to compare the errors in different methods. The self-similar solution also acts as an attractor for general solutions.

In chapter three, I shall look at three different static mesh methods used to solve the partial differential equation in conservation form, based on the standard Crank-Nicolson scheme. First, I will linearise the equation in u^{n+1} and use a standard solver. I will then look at two iterative ways of solving the full non-linear equations, using first Picard iteration and then the Newton-Raphson method.

In chapter four, I will look at solutions of the PDE with $n = 4$ in non-conservation form, as in

$$\frac{\partial u}{\partial t} = u^4 \frac{\partial^2 u}{\partial x^2} + 4u^3 \left(\frac{\partial u}{\partial x}\right)^2 ,$$

which comes from (1.2) by direct differentiation. In this form, I will use the Crank-Nicolson scheme and compare methods using upwind and central differences for the last term.

In chapter five I will look at a case where the diffusion coefficient $D(u)$ in (1.1) is not known analytically but is tabulated for a series of values of u. I will investigate two ways that can be used to extend the above methods for use in this case. First, I will simply use linear interpolation to find the value $D(u)$ for a value u not present in the table. Second, I will use a least squares minimisation to fit a function to the tabular data and then use this function to evaulate the different methods.

In chapter six I will look at a moving mesh method for solving the problem. Here, the area under the graph between consecutive grid-points is kept the same throughout the problem and this fact is used to calculate the new grid points for each timestep. An ordinary differential equation is formed that gives the grid points at the next time step and this is solved using two different methods, fourth order Runge-Kutta and second order backward differentiation with the initial values given by second order Runge-Kutta.

In chapter seven I give the conclusions from this project and also look at further work, including the use of the Newton-Krylov method for solving schemes for equation (1.1).

2 A Self-Similar Solution

2.1 Scale Invariance

First we seek a coordinate transform under which the partial differential equation (1.6) is invariant. Consider the scaling transformation

$$u = \lambda^{\gamma}\hat{u} \qquad t = \lambda\hat{t} \qquad x = \lambda^{\beta}\hat{x} \tag{2.1}$$

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

Transforming the terms of the PDE into the variables $(\hat{u}, \hat{t}, \hat{x})$, we have

$$\frac{\partial u}{\partial t} = \frac{\lambda^{\gamma}}{\lambda}\frac{\partial \hat{u}}{\partial \hat{t}}$$

and

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

Transforming the right hand side of the PDE gives

$$\frac{\partial}{\partial x}\left(u^4\frac{\partial u}{\partial x}\right) = \frac{1}{\lambda^{\beta}}\frac{\partial}{\partial \hat{x}}\left(\lambda^{4\gamma}\hat{u}^4\frac{\lambda^{\gamma}}{\lambda^{\beta}}\frac{\partial \hat{u}}{\partial \hat{x}}\right) = \frac{\lambda^{5\gamma}}{\lambda^{2\beta}}\frac{\partial}{\partial \hat{x}}\left(\hat{u}^4\frac{\partial \hat{u}}{\partial \hat{x}}\right).$$

Hence our transformed PDE is

$$\frac{\lambda^{\gamma}}{\lambda}\frac{\partial \hat{u}}{\partial \hat{t}} = \frac{\lambda^{5\gamma}}{\lambda^{2\beta}}\frac{\partial}{\partial \hat{x}}\left(\hat{u}^4\frac{\partial \hat{u}}{\partial \hat{x}}\right).$$

Therefore, for equation (1.6) to be invariant under the transform (2.1), we require that

$$\gamma - 1 = 5\gamma - 2\beta. \tag{2.2}$$

To determine γ and β we need another equation.

Integrating equation (1.6) gives

$$\int_a^b \frac{\partial u}{\partial t} dx = \int_a^b \frac{\partial}{\partial x}\left(u^4 \frac{\partial u}{\partial x}\right) dx.$$

This simplifies to become

$$\frac{d}{dt}\int_a^b u dx = u^4 \frac{\partial u}{\partial x}\bigg|_a^b .$$

Taking the boundary conditions to be $u(a) = u(b) = 0$, we get

$$\frac{d}{dt}\int_a^b u dx = 0$$

and hence

$$\int_a^b u dx = k \tag{2.3}$$

where k is a constant. Thus the integral of the solution is conserved.

Transforming this integral to the variables $(\hat{u}, \hat{t}, \hat{x})$, we obtain

$$\int_a^b \lambda^\gamma \hat{u} \lambda^\beta \partial \hat{x} = k$$

which becomes

$$\lambda^{\gamma+\beta} \int_a^b \hat{u} \partial \hat{x} = \lambda^0 k.$$

For the conservation law (2.3) to be invariant under the transformation 2.1, we therefore require

$$\lambda + \beta = 0. \tag{2.4}$$

Solving equations (2.2) and (2.4) simultaneously, we find that

$$\beta = \frac{1}{6} \qquad \gamma = -\frac{1}{6} \tag{2.5}$$

and the scale-invariant transformation becomes

$$u = \lambda^{-\frac{1}{6}} \hat{u} \qquad t = \lambda \hat{t} \qquad x = \lambda^{\frac{1}{6}} \hat{x}. \tag{2.6}$$

2.2 Self-Similar Solutions

A time dependent phenomenon is called self-similar if the spatial distributions of its properties at different times can be obtained from one another by a similarity transform [8].

Note from (2.1) that

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

We now introduce two new variables

$$\phi = \frac{u}{t^{\lambda}} = \frac{\hat{u}}{\hat{t}^{\lambda}}$$

and

$$y = \frac{x}{t^{\beta}} = \frac{\hat{x}}{\hat{t}^{\beta}}.$$

Note that these two variables are independent of λ and are invariant under the transformation (2.1).

Now take ϕ to be a function of y and transform the PDE (1.6) into the variables ϕ and y to obtain an ODE. First, transform the left hand side of the PDE into ϕ and y.

$$\begin{aligned}
\frac{\partial u}{\partial t} &= \frac{\partial}{\partial t}(\phi t^{\gamma}) \\
&= t^{\gamma}\frac{\partial \phi}{\partial t} + \phi\gamma t^{\gamma-1} \\
&= t^{\gamma}\frac{d\phi}{dy}\frac{\partial y}{\partial t} + \phi\gamma t^{\gamma-1} \\
&= t^{\gamma}\frac{d\phi}{dy}\left(\frac{-\beta x}{t^{\beta+1}}\right) + \phi\gamma t^{\gamma-1} \\
&= -\frac{\beta y}{t}t^{\gamma}\frac{d\phi}{dy} + \phi\gamma t^{\gamma-1} \\
&= -\beta t^{\gamma-1}y\frac{d\phi}{dy} + \gamma\phi t^{\gamma-1}
\end{aligned}$$

Now transform the right hand side into ϕ and y.

$$\begin{aligned}
\frac{\partial}{\partial x}\left(u^4\frac{\partial u}{\partial x}\right) &= \frac{\partial y}{\partial x}\frac{d}{dy}\left(\phi^4 t^{4\gamma}\frac{\partial y}{\partial x}\frac{\partial u}{\partial \phi}\frac{d\phi}{dy}\right)\\
&= \frac{1}{t^\beta}\frac{d}{dy}\left(\phi^4 t^{4\gamma}t^{\gamma-\beta}\frac{d\phi}{dy}\right)\\
&= t^{5\gamma-2\beta}\frac{d}{dy}\left(\phi^4\frac{d\phi}{dy}\right)
\end{aligned}$$

Substituting these into equation (1.6) gives

$$-\beta t^{\gamma-1}y\frac{d\phi}{dy}+\gamma\phi t^{\gamma-1}=t^{5\gamma-2\beta}\frac{d}{dy}\left(\phi^4\frac{d\phi}{dy}\right).$$

Setting $\gamma=-\frac{1}{6}$ and $\beta=\frac{1}{6}$ as given in equation (2.5) gives

$$-\frac{y}{6}\frac{d\phi}{dy}-\frac{\phi}{6}=\frac{d}{dy}\left(\phi^4\frac{d\phi}{dy}\right).$$

This is now an ODE for the function $\phi(y)$,

$$\frac{\phi}{6}+\frac{y}{6}\frac{d\phi}{dy}+\frac{d}{dy}\left(\phi^4\frac{d\phi}{dy}\right)=0. \qquad (2.7)$$

This equation can now be solved to find the self-similar solution. Equation (2.7) can be rearranged to give

$$\frac{d}{dy}\left(\phi^4\frac{d\phi}{dy}\right)+\frac{1}{6}\frac{d}{dy}(y\phi)=0.$$

Integrating this gives

$$\phi^4\frac{d\phi}{dy}+\frac{1}{6}y\phi=C$$

where C is a constant of integration. Taking C to be equal to 0 assuming that $\frac{d\phi}{dy}=0$ when $\phi=0$ gives

$$\phi^3\frac{d\phi}{dy}+\frac{1}{6}y=0.$$

Rearranging gives

$$\int \phi^3 d\phi = -\frac{1}{6}\int y dy.$$

This integrates to give

$$\frac{\phi^4}{4} = -\frac{1}{6}\left(\frac{y^2}{2} - d\right),$$

where d is a constant of integration. This simplifies to

$$\phi = \left(\frac{2}{3}\right)^{\frac{1}{4}}\left(d - \frac{y^2}{2}\right)^{\frac{1}{4}}.$$

where d is constant. Since ϕ has to be positive we obtain

$$\phi = \begin{cases} (A - \frac{1}{3}y^2)^{\frac{1}{4}} & \frac{1}{3}y^2 \leqslant A \\ 0 & \frac{1}{3}y^2 > A \end{cases}$$

which we write as

$$\phi = \left(A - \frac{1}{3}y^2\right)_+^{\frac{1}{4}}.$$

Mapping this back to the original variables (u,x,t), we obtain the solution

$$u = \frac{1}{t^{\frac{1}{6}}}\left(A - \frac{1}{3}\left(\frac{x}{t^{\frac{1}{6}}}\right)^2\right)_+^{\frac{1}{4}}.$$

This is a self-similar solution for the original partial differential equation. Figure 9.2 from [3] shows a self-similar solution at two points in time and is reproduced here as figure 1. The transformation given in equation (2.6) can be used to transform the solution at time t_1 onto the solution at time t_2. The value of λ used in the transformation will depend on t_1 and t_2.

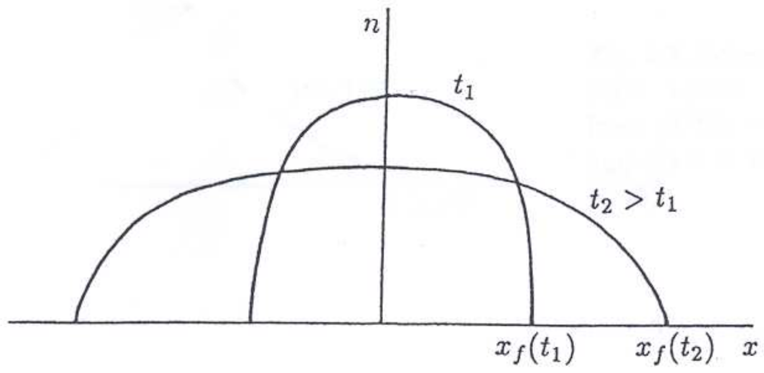

Figure 1: Self-Similar Solution at times t_1 and t_2

2.3 The Self-Similar Solution as an Attractor

Comparison theory says that if we have two solutions u, w to the problem with $u \geqslant w$ at t_0 then $u \geqslant w$ for all time, giving an ordering of solutions [4]. The self-similar solution is of particular interest due to the following convergence result from [5].

"Let $u(x,t) \geqslant 0$ be an arbitrary solution of equation (1.6) with integral I and centre of mass x_0. Then if $\bar{u}(x;t;A)$ is the self-similar solution with the same integral and centre of mass, then for all t_0 we have

$$t^{\frac{1}{3}} \parallel u - \bar{u} \parallel_{L_1} \to 0 \qquad \text{as} \qquad t \to \infty.$$

Equivalently, the PDE (1.6) has as a global attractor the solution of the ODE (2.7) with the same first integral." This means that a solution with arbitrary initial data will be squeezed between two discrete self-similar solutions. Diagram 2 from [4] show this for an initial solution that fits between self-similar solutions with $A = 0.9$ and $A = 2.3$. The figures in this diagram are plotted in a reference space ξ.

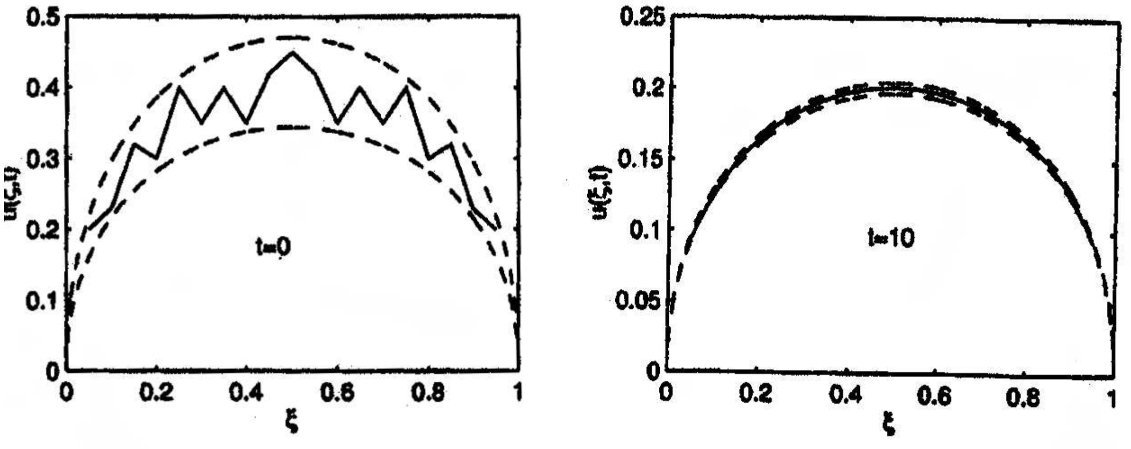

Figure 2: Convergence of the Solution in the Computational Domain

3 Solutions on a Stationary Mesh - Conservation Form

The partial differential equation will be first solved numerically using finite differences applied using the theta method, a weighted average of a fully implicit and a fully explicit method.

Where ν is used in the following sections, it is a constant and is equal to $\dfrac{\Delta t}{\Delta x^2}$ unless specified otherwise.

Assuming that $u(a) = u(b) = 0$, the PDE in equation (1.6) is in conservation form. This means that the area under the solution curve is conserved. The numerical solution should also conserve this area. Using standard finite differences gives

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} =$$

$$\frac{\theta}{\Delta x^2}\left(\left(\frac{u_{j+1}^{n+1} + u_j^{n+1}}{2}\right)^4 \left(u_{j+1}^{n+1} - u_j^{n+1}\right) - \left(\frac{u_j^{n+1} + u_{j-1}^{n+1}}{2}\right)^4 \left(u_j^{n+1} - u_{j-1}^{n+1}\right)\right)$$

$$+\frac{(1-\theta)}{\Delta x^2}\left(\left(\frac{u_{j+1}^n + u_j^n}{2}\right)^4 \left(u_{j+1}^n - u_j^n\right) - \left(\frac{u_j^n + u_{j-1}^n}{2}\right)^4 \left(u_j^n - u_{j-1}^n\right)\right).$$

There are several different methods for coping with the non-linear implicit part of the equation and these will be investigated in the next sections.

3.1 Crank-Nicolson with Explicit Treatment of u^4

In this section, the non-linear term of the equation will be taken at time level n. This results in the semi-implicit scheme

$$\frac{u_j^{n+1}-u_j^n}{\Delta t} =$$

$$\frac{\theta}{\Delta x^2}\left(\left(\frac{u_{j+1}^n+u_j^n}{2}\right)^4\left(u_{j+1}^{n+1}-u_j^{n+1}\right)-\left(\frac{u_j^n+u_{j-1}^n}{2}\right)^4\left(u_j^{n+1}-u_{j-1}^{n+1}\right)\right)$$

$$+\frac{(1-\theta)}{\Delta x^2}\left(\left(\frac{u_{j+1}^n+u_j^n}{2}\right)^4\left(u_{j+1}^n-u_j^n\right)-\left(\frac{u_j^n+u_{j-1}^n}{2}\right)^4\left(u_j^n-u_{j-1}^n\right)\right). \tag{3.1}$$

After rearranging, this becomes

$$u_{j-1}^{n+1}\left(-\nu\theta\left(\frac{u_j^n+u_{j-1}^n}{2}\right)^4\right)+u_j^{n+1}\left(1+\nu\theta\left(\left(\frac{u_{j+1}^n+u_j^n}{2}\right)^4+\left(\frac{u_j^n+u_{j-1}^n}{2}\right)\right)\right)+$$

$$u_{j+1}^{n+1}\left(-\nu\theta\left(\frac{u_{j+1}^n+u_j^n}{2}\right)^4\right)=u_j^n+$$

$$\nu(1-\theta)\left(\left(\frac{u_{j+1}^n+u_j^n}{2}\right)^4\left(u_{j+1}^n-u_j^n\right)-\left(\frac{u_j^n+u_{j-1}^n}{2}\right)^4\left(u_j^n-u_{j-1}^n\right)\right). \tag{3.2}$$

This is solved as a matrix equation $A\mathbf{u}^{n+1}=\mathbf{b}$ with A being the matrix of coefficients of the components of the vector $\mathbf{u}^{n+1}$ and $\mathbf{b}$ being a vector of the right hand side of equation (3.2). This can easily be solved using any standard method. In this case, A is tridiagonal and we use the Thomas algorithm [11].

3.1.1 Stability and Accuracy

The stability of this method can be examined using Fourier analysis. Consider the diffusion coefficient u^4 to be frozen and replace it with a constant. The PDE now becomes

$$\frac{\partial u}{\partial t} = \sigma \frac{\partial^2 u}{\partial x^2},$$

where σ is constant.

The scheme to be analysed is simply

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{\sigma}{\Delta x^2}\left(\theta\left(u_{j+1}^{n+1} - 2u_j^{n+1} - u_{j-1}^{n+1}\right) + (1-\theta)\left(u_{j+1}^n - 2u_j^n - u_{j-1}^n\right)\right). \tag{3.3}$$

Substituting $u_j^n = a_n e^{ikj\Delta x}$ into the equation (3.3), multiplying through by Δt and dividing through by $e^{ijk\Delta x}$ gives

$$a_{n+1} - a_n = \nu\left((1-\theta)a_n + \theta a_{n+1}\right)\left(e^{ik\Delta x} - 2 + e^{-ik\Delta x}\right),$$

where $\nu = \frac{\sigma \Delta t}{\Delta x^2}$.

This then becomes

$$a_{n+1} = \left(\frac{1 - 4\nu(1-\theta)\sin^2\left(\frac{k\Delta x}{2}\right)}{1 + 4\nu\theta \sin^2\left(\frac{k\Delta x}{2}\right)}\right) a_n.$$

For stability, we require that $|a_{n+1}| \leqslant |a_n|$. For this, we need

$$-1 - 4\nu\theta\sin^2\left(\frac{k\Delta x}{2}\right) \leqslant 1 - 4\nu(1-\theta)\sin^2\left(\frac{k\Delta x}{2}\right) \leqslant 1 + 4\nu\theta\sin^2\left(\frac{k\Delta x}{2}\right).$$

The right hand part of this inequality always holds. Moreover, for $\theta \geqslant \frac{1}{2}$, the left hand side of the equation is always $\leqslant 0$ so the inequality holds and the scheme is unconditionally stable.

If $\theta < \frac{1}{2}$, then the scheme will be stable only if $\nu(1-2\theta)\sin^2\left(\frac{k\Delta x}{2}\right) \leqslant \frac{1}{2}$. The worst possible case is when $\sin^2\left(\frac{k\Delta x}{2}\right) = 1$. This gives Fourier stability for $\nu \leqslant \frac{1}{2(1-2\theta)}$.

If θ is taken to be equal to $\frac{1}{2}$ then the method is known as the Crank-Nicolson method. This method is stable for all ν and is second order accurate in both space and time. The stability can be exploited to use large timesteps (with the order of the space and time steps being equal) and since it is second order accurate in time, good accuracy will still be obtained.

σ is obtained by freezing u^4 so the maximum value it can take is the maximum value of u^4. Due to the maximum principle, this is equal to the maximum value of u^4 in the initial data. Hence the maximum possible value of ν is $\frac{\Delta t}{\Delta x^2} \max_i u_i^4$.

3.1.2 A Maximum Principle

The theta method of equation (3.1) with $0 \leqslant \theta \leqslant 1$ and

$$\nu(1-\theta) \leqslant \frac{1}{2} \tag{3.4}$$

yields u_j^n satisfying

$$u_{min} \leqslant u_j^n \leqslant u_{max}$$

where

$$u_{min} := \min\{u_0^m, 0 \leqslant m \leqslant n; u_j^0, 0 \leqslant j \leqslant J; u_J^m, 0 \leqslant m \leqslant n\},$$

and

$$u_{max} := \max\{u_0^m, 0 \leqslant m \leqslant n; u_j^0, 0 \leqslant j \leqslant J; u_J^m, 0 \leqslant m \leqslant n\}$$

are the minimum and maximum values of u on the initial line and the boundaries [2].

Using this maximum principle we can deduce stability and hence convergence. For any ν which satisfies the stability condition (3.4), the approximations given by equation (3.1) with consistent initial and Dirichlet boundary data converge uniformly if the initial data are smooth enough for the truncation error to tend to zero as Δt and Δx are decreased, whilst keeping ν constant [2].

The condition used in the above theorem, $\nu(1-\theta) \leqslant \dfrac{1}{2}$ is much more restrictive than that obtained using Fourier stability analysis, $\nu(1-2\theta) \leqslant \dfrac{1}{2}$. For example, the Crank-Nicolson scheme is always stable but only if $\nu \leqslant 1$ does it satisify the maximum principle which is then used to deduce stability and convergence. If the boundary conditions are $u_0^n = u_J^n = 0$ then we want

$$\left|u_j^n\right| \leqslant K \max_{0 \leqslant i \leqslant J} \left|u_i^0\right| \qquad \forall j, n$$

to be satisfied with $K = 1$ for a maximum principle to hold. However, for Fourier stability any value of K is accepted in this bound. The weaker condition $\nu(1-2\theta) \leqslant \dfrac{1}{2}$ is then adequate [2].

Hence the maximum priciple can be viewed as an alternative way of obtaining stability conditions but it may derive conditions that are only sufficient.

3.2 Crank-Nicolson with Semi-Implicit Treatment of u^4

3.2.1 Picard Iteration

Here, the non-linear term of the implicit section of equation is taken at time level $n+\frac{1}{2}$. It is known that for the Crank-Nicolson method, the diffusion coefficient, here u^4, produces better accuracy if taken at time level $n+\frac{1}{2}$. At the start of each time step, u^4 is taken at time level n and the problem is iterated to find a provisional value of u^{n+1}. Then $u^{n+\frac{1}{2}}$ is calculated as $\dfrac{u^{n+1}+u^n}{2}$ and the process repeated. When the solution for u^{n+1} has converged, the time step is advanced and the next time step commenced. The scheme is

$$\frac{u_j^{n+1}-u_j^n}{\Delta t} =$$

$$\frac{\theta}{\Delta x^2}\left(\left(\frac{u_{j+1}^{n+\frac{1}{2}}+u_j^{n+\frac{1}{2}}}{2}\right)^4\left(u_{j+1}^{n+1}-u_j^{n+1}\right)\left(\frac{u_j^{n+\frac{1}{2}}+u_{j-1}^{n+\frac{1}{2}}}{2}\right)^4\left(u_j^{n+1}-u_{j-1}^{n+1}\right)\right)$$

$$+\frac{(1-\theta)}{\Delta x^2}\left(\left(\frac{u_{j+1}^n+u_j^n}{2}\right)^4\left(u_{j+1}^n-u_j^n\right)-\left(\frac{u_j^n+u_{j-1}^n}{2}\right)^4\left(u_j^n-u_{j-1}^n\right)\right).$$

The equations can be solved using Picard iteration with the matrix equation being solved as before.

3.2.2 Newton-Raphson Iteration

Here, the non-linear part of the implicit section of the method is taken at time level $n+1$.

Let

$$F_j\left(\mathbf{u}^{n+1}\right) = 0 = u_j^{n+1} - u_j^n -$$

$$\frac{\Delta t}{\Delta x^2}\left(\left(\frac{u_{j+1}^{n+1}+u_j^{n+1}}{2}\right)^4\left(u_{j+1}^{n+1}-u_j^{n+1}\right)-\left(\frac{u_j^{n+1}+u_{j-1}^{n+1}}{2}\right)^4\left(u_j^{n+1}-u_{j-1}^{n+1}\right)\right).$$

To solve the equation $\mathbf{F}\left(\mathbf{u}^{n+1}\right)=\mathbf{0}$, this method iterates

$$\begin{aligned}\mathbf{J}(\mathbf{u}^{n+1})^p\delta\mathbf{u} &= -\mathbf{F}(\mathbf{u}^{n+1})^p \\ \left(\mathbf{u}^{n+1}\right)^{p+1} &= \left(\mathbf{u}^{n+1}\right)^p + \delta\mathbf{u}\end{aligned} \tag{3.5}$$

where p is the iteration count and $\mathbf{J}$ is the jacobian matrix $\dfrac{\partial\mathbf{F}(\mathbf{u})}{\partial\mathbf{u}}$. The Jacobian is evaluated by analytically finding the derivatives of $\mathbf{F}(\mathbf{u})$ and coding these into the fortran. The Jacobian is tridiagonal so the equation can be solved using the Thomas algorithm as before.

At the start of each time step, u^4 is taken at time level n and Newton-Raphson is applied as in (3.5) to find u^{n+1}. When the solution for u^{n+1} has converged, the time step is advanced and the process repeated.

If the iteration in equations (3.5) can be rearranged to the form $(\mathbf{u}^{n+1})^{p+1} = \mathbf{C}(\mathbf{u}^{n+1})^p$ then the sequence generated by the iteration will converge quadratically when the eigenvalues of $\mathbf{C}$ are all less than one in modulus. In practice, this condition is very hard to find and the convergence condition must be found by numerical experiment.

3.3 Results

3.3.1 Evolution of Various Initial Conditions

Figure 3 shows the evolution of the solution with self-similar initial data

$$u = \begin{cases} \left(1 - \frac{x^2}{3}\right)^{\frac{1}{4}} & \text{for } x^2 < 3 \\ 0 & \text{for } x^2 \geqslant 3 \end{cases}$$

using the explicit scheme with u^4 at time level n.

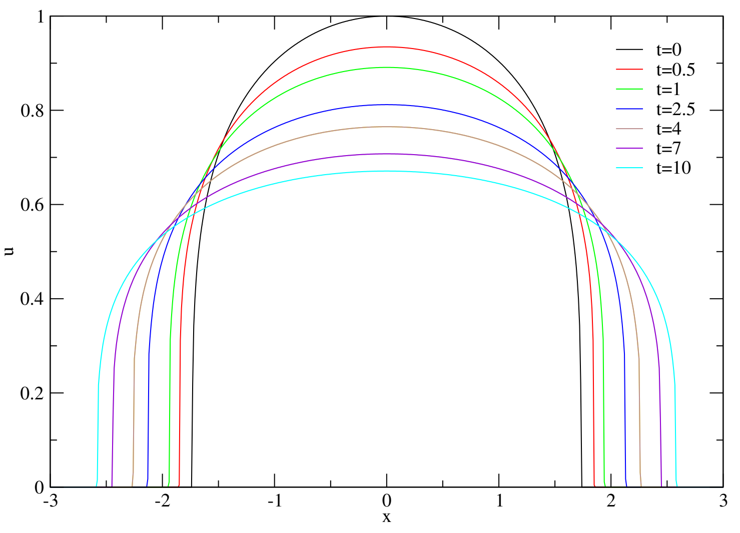

Figure 3: Evolution of solution from explicit scheme with initial condition $u = \left(1 - \frac{x^2}{3}\right)^{\frac{1}{4}}$

Figure 4 shows the evolution of the solution with non-self-similar initial data

$$u = \begin{cases} \cos\left(\frac{\pi x}{2}\right) & \text{for } |x| < 1 \\ 0 & \text{for } |x| \geqslant 1 \end{cases}$$

using the implicit scheme with Picard iteration.

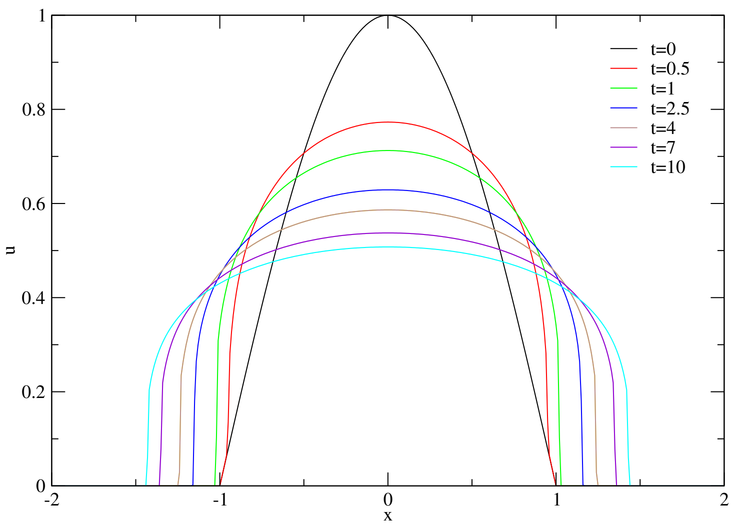

Figure 4: Evolution of solution from implicit scheme with Picard iteration and initial condition $u = \cos\left(\frac{\pi x}{2}\right)$

Figure 5 shows the evolution of the solution with perturbed non-self-similar initial data

$$u = \begin{cases} \cos\left(\frac{\pi x}{2}\right) + 0.2\cos\left(\frac{21\pi x}{2}\right) & \text{for } |x| < 1 \\ 0 & \text{for } |x| \geqslant 1 \end{cases}$$

using the implicit scheme with Newton Raphson iteration.

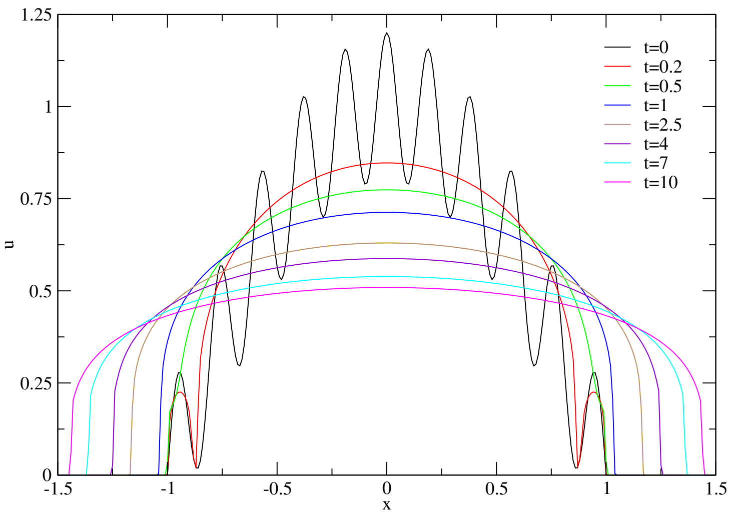

Figure 5: Evolution of solution from implicit scheme with Newton-Raphson iteration and initial condition $u = \cos\left(\frac{\pi x}{2}\right) + 0.2\cos\left(\frac{21\pi x}{2}\right)$

3.3.2 The Self-Similar Solution as an Attractor for the Numerical Solution

The self-similar solution is known analytically to be an attractor for any other solution but it is unknown whether this will be true for the solution found using the numerical scheme. Figures 6 and 7 show the results from the explicit implementation of u^4 when applied to two different initial conditions, each taking two self-similar solutions that sandwich the initial condition at the initial time. Similar results are obtained using the other schemes.

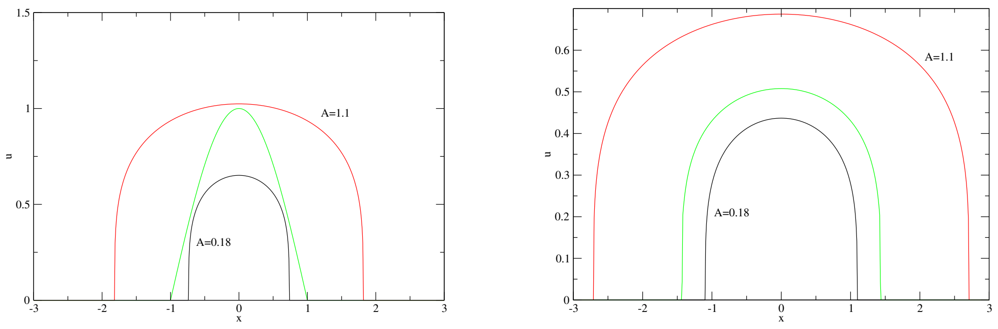

Figure 6: Initial Condition $u = \cos\left(\frac{\pi x}{2}\right)$ sandwiched by two self-similar solutions. Results from explicit method.

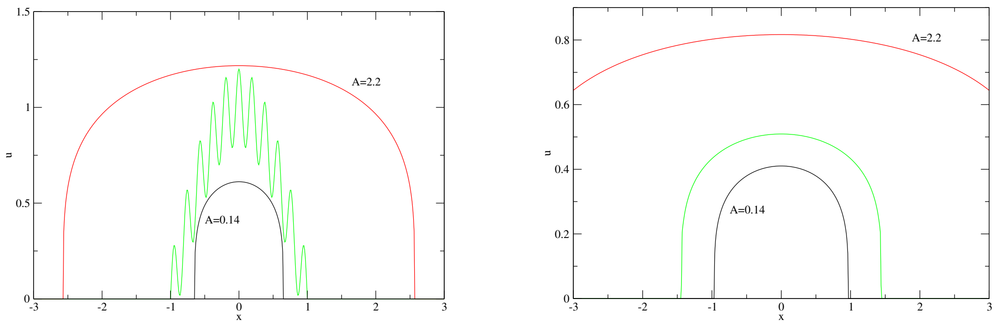

Figure 7: Initial Condition $u = \cos\left(\frac{\pi x}{2}\right) + 0.2\cos\left(\frac{21\pi x}{2}\right)$ sandwiched by two self-similar solutions. Results from explicit method.

4 Solutions on a Stationary Mesh - Non-Conservation Form

The partial differential equation will now be solved numerically in non-conservation form using finite differences applied using the theta method. Where ν is used in the following section, it is a constant and is equal to $\dfrac{\Delta t}{\Delta x^2}$.
The PDE (1.6) can be differentiated to give

$$\frac{\partial u}{\partial t} = u^4 \frac{\partial^2 u}{\partial x^2} + 4u^3 \left(\frac{\partial u}{\partial x}\right)^2 .$$

Finite differences can be applied to this equation in two different ways.

4.1 Upwind Finite Differencing

Again, the non-linear parts of this equation are taken always at time level n. Diffusion occurs always outwards from the origin so upwind finite differences are chosen for $\dfrac{\partial u}{\partial x}$, these being different for $x > 0$ and $x < 0$. The scheme is

$x > 0$:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{\theta}{\Delta x^2} \left((u_j^n)^4 \left(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}\right) + 4(u_j^n)^3 \left(u_j^n - u_{j-1}^n\right)\left(u_j^{n+1} - u_{j-1}^{n+1}\right)\right)$$

$$+\frac{(1-\theta)}{\Delta x^2} \left((u_j^n)^4 \left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right) + 4(u_j^n)^3 \left(u_j^n - u_{j-1}^n\right)^2\right)$$

$x < 0$:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{\theta}{\Delta x^2} \left((u_j^n)^4 \left(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}\right) + 4(u_j^n)^3 \left(u_{j+1}^n - u_j^n\right)\left(u_{j+1}^{n+1} - u_j^{n+1}\right)\right)$$

$$+\frac{(1-\theta)}{\Delta x^2}\left((u_j^n)^4\left(u_{j+1}^n-2u_j^n+u_{j-1}^n\right)+4(u_j^n)^3\left(u_{j+1}^n-u_j^n\right)^2\right)$$

At $x=0$, either case can be used.

4.2 Central Differencing

Central differences are of a higher order than upwind differences but they take data from a wider area and so may use data on which the solution is not physically dependent. This may cause the solution to be less physical than the upwind solution. The central difference scheme is

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}=$$

$$\frac{\theta}{\Delta x^2}\left((u_j^n)^4\left(u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1}\right)+(u_j^n)^3\left(u_{j+1}^n-u_{j-1}^n\right)\left(u_{j+1}^{n+1}-u_{j-1}^{n+1}\right)\right)$$

$$+\frac{(1-\theta)}{\Delta x^2}\left((u_j^n)^4\left(u_{j+1}^n-2u_j^n+u_{j-1}^n\right)+(u_j^n)^3\left(u_{j+1}^n-u_{j-1}^n\right)^2\right).$$

4.3 Results

4.3.1 Evolution of Various Initial Conditions

Figure 8 shows the evolution of the solution with perturbed non-self-similar initial data

$$u=\begin{cases}\cos\left(\frac{\pi x}{2}\right)+0.2\cos\left(\frac{21\pi x}{2}\right) & \text{for } |x|<1\\ 0 & \text{for } |x|\geqslant 1\end{cases}$$

using the scheme with upwind differencing.

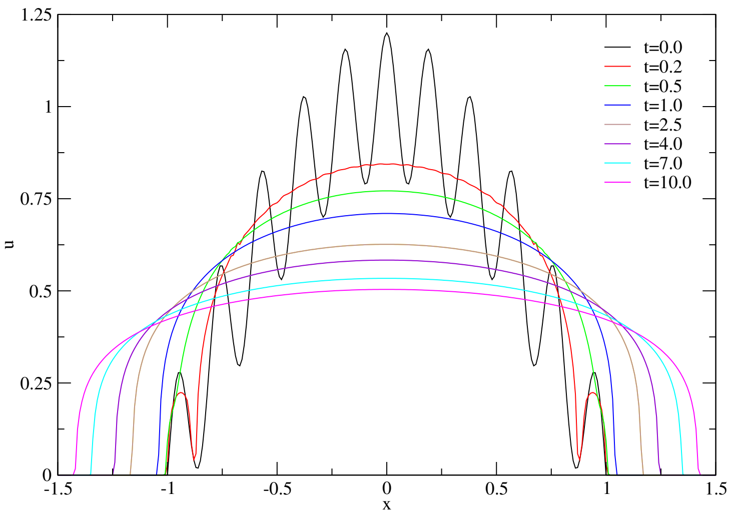

Figure 8: Evolution of solution from upwind difference scheme with initial condition $u = \cos\left(\frac{\pi x}{2}\right) + 0.2\cos\left(\frac{21\pi x}{2}\right)$

Figure 9 shows the evolution of the solution with non-self-similar initial data

$$u = \begin{cases} \cos\left(\frac{\pi x}{2}\right) & \text{for } |x| < 1 \\ 0 & \text{for } |x| \geqslant 1 \end{cases}$$

using the scheme with central differencing.

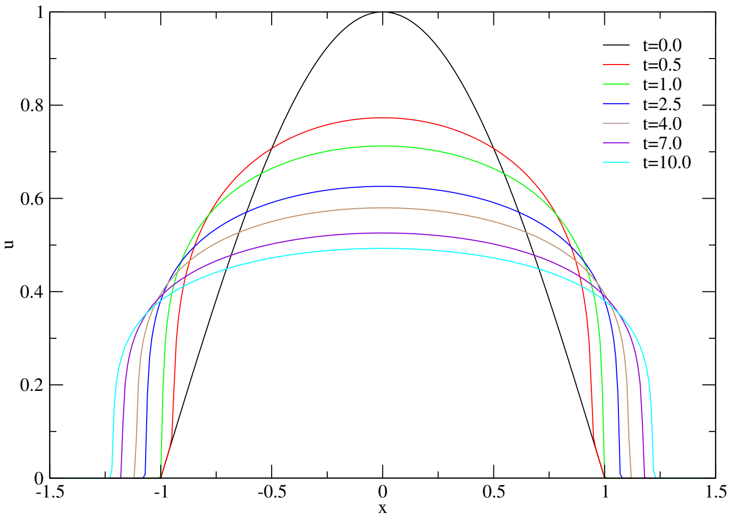

Figure 9: Evolution of solution from central difference scheme with initial condition $u = \cos\left(\frac{\pi x}{2}\right)$

5 Solutions on a Stationary Mesh - A Tabular Non-Linearity

For some applications, the exact form of the diffusion coefficient, $D(u)$ in equation (1.1) is not known in analytic form. Instead, a table is available giving $D(u)$ for a set of values of u. The methods used in the previous two chapters can be applied to these circumstances in various different ways. Two methods for extending the previous schemes (using an analytical form of coefficient) to using a tabular coefficient will be investigated. The first will involve a straightforward linear interpolation to find the value of $D(u)$ for a given value of u. The second will look at using a least squares minimisation method to fit a polynomial to the data given in the table and then using this polynomial in the schemes. This second method will require the least change to the schemes as it simply requires u^4 to be replaced by a more general polynomial.

5.1 Linear Interpolation

If the value of D is required at a value u^* then the closest tabulated values, u_1 and u_2 of u on either side of u^* must first be found. This is straightforward to achieve by a simple search. The linear interpolation function D_{LI} can then be used to find an approximation to $D(u^*)$ as follows,

$$D\left(u^*\right) \simeq D_{LI}\left(u^*\right) = \left(\frac{u_2 - u^*}{u_2 - u_1}\right) D_{LI}\left(u_1\right) + \left(\frac{u^* - u_1}{u_2 - u_1}\right) D_{LI}\left(u_2\right).$$

Hence, for any value of u^*, $D(u^*)$ can be estimated and it is possible to apply this to the methods in the previous two chapters. If the value u^* is outside the range of the table then linear extrapolation is carried out using the final two points in the table.

5.1.1 Conservation Form - Explicit Implementation of u^4

This scheme is easy to convert to use the tabular data. The scheme becomes

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}=$$

$$\frac{\theta}{\Delta x^2}\left(D_{LI}\left(\frac{u_{j+1}^n+u_j^n}{2}\right)\left(u_{j+1}^{n+1}-u_j^{n+1}\right)-D_{LI}\left(\frac{u_j^n+u_{j-1}^n}{2}\right)\left(u_j^{n+1}-u_{j-1}^{n+1}\right)\right)$$

$$+\frac{1-\theta}{\Delta x^2}\left(D_{LI}\left(\frac{u_{j+1}^n+u_j^n}{2}\right)\left(u_{j+1}^n-u_j^n\right)-D_{LI}\left(\frac{u_j^n+u_{j-1}^n}{2}\right)\left(u_j^n-u_{j-1}^n\right)\right).$$

This can be rearranged and solved as in Section 3.1.

5.1.2 Conservation Form - Semi-Implicit Implementation of u^4 - Picard Iteration

Similarly to the previous scheme, it is easy to change this scheme to use the tabular data. All the details of the scheme and how it is implemented are as in Section 3.2.1 and the equation to be iterated is as follows,

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}=$$

$$\frac{\theta}{\Delta x^2}\left(D_{LI}\left(\frac{u_{j+1}^{n+\frac{1}{2}}+u_j^{n+\frac{1}{2}}}{2}\right)\left(u_{j+1}^{n+1}-u_j^{n+1}\right)-D_{LI}\left(\frac{u_j^{n+\frac{1}{2}}+u_{j-1}^{n+\frac{1}{2}}}{2}\right)\left(u_j^{n+1}-u_{j-1}^{n+1}\right)\right)$$

$$+\frac{1-\theta}{\Delta x^2}\left(D_{LI}\left(\frac{u_{j+1}^n+u_j^n}{2}\right)\left(u_{j+1}^n-u_j^n\right)-D_{LI}\left(\frac{u_j^n+u_{j-1}^n}{2}\right)\left(u_j^n-u_{j-1}^n\right)\right).$$

5.1.3 Conservation Form - Semi-Implicit Implementation of u^4 - Newton-Raphson Iteration

This method is more complicated to convert to use tabular data due to the need to form the Jacobian of $\mathbf{F}\left(\mathbf{u}^{n+1}\right)$ as given in (5.1).

$$\mathbf{F}_j\left(\mathbf{u}^{n+1}\right)=u_j^{n+1}-u_j^n-$$

$$\nu\left(D_{LI}\left(\frac{u_{j+1}^{n+1}+u_j^{n+1}}{2}\right)\left(u_{j+1}^{n+1}-u_j^{n+1}\right)-D_{LI}\left(\frac{u_j^{n+1}+u_{j-1}^{n+1}}{2}\right)\left(u_j^{n+1}-u_{j-1}^{n+1}\right)\right). \tag{5.1}$$

The Jacobian can no longer be found analytically in full, parts of it must be found numerically. Taking the diagonal entry for example, we get

$$\frac{\partial F_j}{\partial u_j^{n+1}}=$$

$$1-\nu\left(-D_{LI}\left(\frac{u_{j+1}^{n+1}+u_j^{n+1}}{2}\right)+\left(u_{j+1}^{n+1}-u_j^{n+1}\right)\frac{\partial D_{LI}}{\partial u_j^{n+1}}\left(\frac{u_{j+1}^{n+1}+u_j^{n+1}}{2}\right)-\right.$$

$$\left.D_{LI}\left(\frac{u_j^{n+1}+u_{j-1}^{n+1}}{2}\right)-\left(u_j^{n+1}-u_{j-1}^{n+1}\right)\frac{\partial D_{LI}}{\partial u_j^{n+1}}\left(\frac{u_j^{n+1}+u_{j-1}^{n+1}}{2}\right)\right).$$

The upper and lower diagonal entries are found similarly. The derivatives are taken as central differences, for example

$$\frac{\partial D_{LI}}{\partial u_j^{n+1}}\left(\frac{u_{j+1}^{n+1}+u_j^{n+1}}{2}\right)=\frac{D_{LI}\left(\frac{u_{j+2}^{n+1}+u_{j+1}^{n+1}}{2}\right)-D_{LI}\left(\frac{u_j^{n+1}+u_{j-1}^{n+1}}{2}\right)}{u_{j+1}^{n+1}-u_{j-1}^{n+1}}.$$

The equation $\mathbf{F}\left(\mathbf{u}^{n+1}\right) = 0$ can now be solved exactly as in Section 3.2.2, using this new implementation of the Jacobian.

5.1.4 Non-Conservation Form - Upwind Finite Differencing

This is slightly more complicated to alter to use the tabular data since the scheme requires that $D_{LI}(u)$ is differentiated. However, an approximation to the derivative is easily achieved using an upwinded numerical derivative. The scheme is given in equations (5.2) and (5.3) for the cases $x > 0$ and $x < 0$ respectively.

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} =$$

$$\frac{\theta}{\Delta x^2}\left(D_{LI}\left(u_j^n\right)\left(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}\right) + \left(u_j^{n+1} - u_{j-1}^{n+1}\right)\left(D_{LI}\left(u_j^n\right) - D_{LI}\left(u_{j-1}^n\right)\right)\right)$$

$$+\frac{1-\theta}{\Delta x^2}\left(D_{LI}\left(u_j^n\right)\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right) + \left(u_j^n - u_{j-1}^n\right)\left(D_{LI}\left(u_j^n\right) - D_{LI}\left(u_{j-1}^n\right)\right)\right) \tag{5.2}$$

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} =$$

$$\frac{\theta}{\Delta x^2}\left(D_{LI}\left(u_j^n\right)\left(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}\right) + \left(u_{j+1}^{n+1} - u_j^{n+1}\right)\left(D_{LI}\left(u_{j+1}^n\right) - D_{LI}\left(u_j^n\right)\right)\right)$$

$$+\frac{1-\theta}{\Delta x^2}\left(D_{LI}\left(u_j^n\right)\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right) + \left(u_{j+1}^n - u_j^n\right)\left(D_{LI}\left(u_{j+1}^n\right) - D_{LI}\left(u_j^n\right)\right)\right) \tag{5.3}$$

5.1.5 Non-Conservation Form - Central Differencing

As with the above method, this requires an approximation of the derivative of $D_{LI}(u)$. The scheme becomes

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} =$$

$$\frac{\theta}{4\Delta x^2}\left(D_{LI}\left(u_j^n\right)\left(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}\right) + \left(u_{j+1}^{n+1} - u_{j-1}^{n+1}\right)\left(D_{LI}\left(u_{j+1}^n\right) - D_{LI}\left(u_{j-1}^n\right)\right)\right)$$

$$+\frac{1-\theta}{4\Delta x^2}\left(D_{LI}\left(u_j^n\right)\left(u_{j+1}^{n} - 2u_j^{n} + u_{j-1}^{n}\right) + \left(u_{j+1}^{n} - u_{j-1}^{n}\right)\left(D_{LI}\left(u_{j+1}^n\right) - D_{LI}\left(u_{j-1}^n\right)\right)\right).$$

5.2 Least Squares Minimisation

In this method, a polynomial function is fitted to the data in the table and this function is used in the place of $D(u)$ in the schemes. The polynomial chosen to be fitted is $Ku^4 + L$. This has been chosen since it is known that for the physical examples considered in chapter one, the function $D(u)$ is generally u^4. This analysis could easily be extended to a more general polynomial.

From [9], the least squares fitting proceeds by finding the sum of the squares of the vertical deviations R^2 of the set of data points (here the points in the table) from the function $\bar{f}$ (here, $Ku_i^4 + L$).

$$R^2 = \sum_{i=1}^{n}\left[D_i - \bar{f}\left(u_i, K, L\right)\right]^2$$

It is now required to minimise these deviations. The condition for R^2 to be a minimum is that

$$\frac{\partial\left(R^2\right)}{\partial K} = 0 \qquad \text{and} \qquad \frac{\partial\left(R^2\right)}{\partial L} = 0. \tag{5.4}$$

In this case, $\bar{f} = Ku_i^4 + L$ so equations (5.4) above become

$$\frac{\partial\left(R^2\right)}{\partial K} = -2\sum_{i=1}^{n}\left[D_i - \left(Ku_i^4 + L\right)\right]u_i^4$$

and

$$\frac{\partial\left(R^2\right)}{\partial L} = -2\sum_{i=1}^{n}\left[D_i - \left(Ku_i^4 + L\right)\right],$$

which leads to the equations

$$L\sum_{i=1}^{n}u_i^4 + K\sum_{i=1}^{n}u_i^8 = \sum_{i=1}^{n}D_iu_i^4$$

and

$$nL + K\sum_{i=1}^{n}u_i^4 = \sum_{i=1}^{n}D_i.$$

In matrix form, this is

$$\begin{bmatrix} L \\ K \end{bmatrix} = \begin{bmatrix} n & \sum\limits_{i=1}^{n}u_i^4 \\ \sum\limits_{i=1}^{n}u_i^4 & \sum\limits_{i=1}^{n}u_i^8 \end{bmatrix}^{-1} \begin{bmatrix} \sum\limits_{i=1}^{n}D_i \\ \sum\limits_{i=1}^{n}u_i^4D_i \end{bmatrix}.$$

Solving for K and L gives

$$L = \frac{\sum\limits_{i=1}^{n}u_i^8\sum\limits_{i=1}^{n}D_i - \sum\limits_{i=1}^{n}u_i^4\sum\limits_{i=1}^{n}u_i^4D_i}{n\sum\limits_{i=1}^{n}u_i^8 - \left(\sum\limits_{i=1}^{n}u_i^4\right)^2}$$

and

$$K = \frac{n\sum_{i=1}^{n} u_i^4 D_i - \sum_{i=1}^{n} u_i^4 \sum_{i=1}^{n} D_i}{n\sum_{i=1}^{n} u_i^8 - \left(\sum_{i=1}^{n} u_i^4\right)^2}.$$

The exact values of K and L are now easily calculated in the fortran program. This diffusion coefficient can easily be applied to all the methods in the previous two chapters, simply by replacing $D(u) = u^4$ by $Ku^4 + L$ and replacing $\frac{\partial D}{\partial u} = 4u^3$ with $4Ku^3$.

5.3 Results

Ordinarily, the data in the table would be experimental data. For the purpose of this project however, this data is not available and the function u^4 has simply been used for $D(u)$. This allows the results from these schemes to be compared to the schemes in the previous two chapters which have u^4 programmed into them.

5.3.1 Least Squares Best Fit

In this case, the result is that $K = 1$ and $L = 0$ so the schemes are exactly the same as those in chapters three and four and the results are identical. The program will run slightly more slowly due to having to calculate K and L at the beginning but this difference is too small to be noticeable. To ensure this scheme works correctly, a new set of data for $D(u)$ have been calculated. For these, the function $D(u) = u^4 + 0.01 \times$ random, where random is a random number, has been used. Running this I obtain $K = 0.9982$ and $L = 0.0055$. The results from this run are shown at the final time $t = 10$ in figure 10 compared to the results obtained from the analytical solution the the PDE (1.6) where $D(u) = u^4$. A noticeable difference in the end results is obvious. This is caused by only a small variation (maximum 1%) in the tabulated data. Since the values in the table are often obtained experimentally, a small error in the experiment may lead to a large change in the nature of the solution.

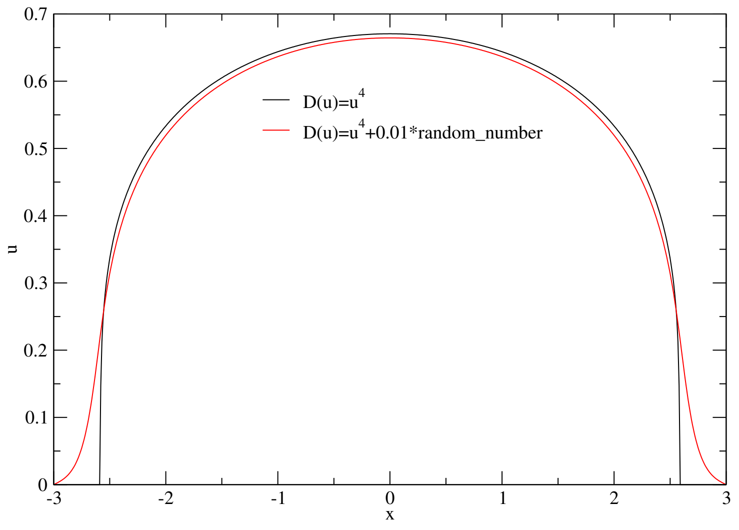

Figure 10: Solution of scheme with $D(u) = u^4 + 0.01 \times \text{random}$ compared to analytic solution with $D(u) = u^4$ at $t = 10$

5.3.2 Linear Interpolation

A comparison to the results of the schemes used in chapters three and four will show the effect of the linear interpolation on the run time and the accuracy of the solution. The results in the table below were obtained using the semi-implicit scheme with Picard iteration.

Method	Δx	Time	Error
Analytic $D(u)$	0.01	$7.65s$	0.0618%
Tabular $D(u)$	0.01	$48.04s$	0.0580%
Analytic $D(u)$	0.25	$0.67s$	0.9125%
Tabular $D(u)$	0.25	$1.60s$	0.9163%

These results show that for suitably small Δx, the accuracy of the solution is not a problem. There is however a large increase in run time. Previously, to find $D(u)$, it was only necessary to find u^4. Now it is necessary to find the correct values to interpolate and then to carry out the interpolation, a much more time consuming task. This increase in run time is magnified as Δx gets smaller and the number of interpolations required to be computed increases. For the method using Newton-Raphson iteration, the increase in run time is even more significant due to the larger number of linear interpolation calls required for each step. The number of linear interpolation calls required for the methods with an explicit implementation of the diffusion coefficient is significantly less than for the implicit methods and hence the increase in run time for these is much smaller.

If the function used for $D(u)$ is changed from u^4 to $u^4 + 0.01 \times \text{random}$, then the evolution of the solution is similar to that seen in the least squares best fit method.

6 A Moving Mesh Method

6.1 Deriving a Moving Mesh Method for the Solution of the Non-Linear Diffusion Equation

With a moving mesh, it must be assumed that the grid points x_i are dependent on time. It can then be shown [1] that

$$\frac{d}{dt}\int_{x_{i-1}(t)}^{x_i(t)} u dx = \int_{x_{i-1}(t)}^{x_i(t)} \frac{\partial u}{\partial t} dx + \int_{x_{i-1}(t)}^{x_i(t)} \frac{\partial}{\partial x}\left(u\frac{dx}{dt}\right) dx.$$

Taking the original partial differential equation $\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(u^4 \frac{\partial u}{\partial x}\right)$ and substituting for $\frac{\partial u}{\partial t}$ in the above equation gives

$$\begin{aligned}\frac{d}{dt}\int_{x_{i-1}(t)}^{x_i(t)} u dx &= \int_{x_{i-1}(t)}^{x_i(t)} \frac{\partial}{\partial x}\left(u^4\frac{\partial u}{\partial x}\right) dx + \int_{x_{i-1}(t)}^{x_i(t)} \frac{\partial}{\partial x}\left(u\frac{dx}{dt}\right) dx \\ &= \int_{x_{i-1}(t)}^{x_i(t)} \frac{\partial}{\partial x}\left(u^4\frac{\partial u}{\partial x} + u\frac{dx}{dt}\right) dx.\end{aligned}$$

Now let $\frac{dx}{dt} = \frac{\partial \phi}{\partial x}$, where ϕ is a velocity potential. The above equation then becomes

$$\begin{aligned}\frac{d}{dt}\int_{x_{i-1}(t)}^{x_i(t)} u dx &= \int_{x_{i-1}(t)}^{x_i(t)} \frac{\partial}{\partial x}\left(u^4\frac{\partial u}{\partial x} + u\frac{\partial \phi}{\partial x}\right) dx \\ &= \int_{x_{i-1}(t)}^{x_i(t)} \frac{\partial}{\partial x}\left(u\left[u^3\frac{\partial u}{\partial x} + \frac{\partial \phi}{\partial x}\right]\right) dx \\ &= \int_{x_{i-1}(t)}^{x_i(t)} \frac{\partial}{\partial x}\left(u\frac{\partial}{\partial x}\left[\frac{u^4}{4} + \phi\right]\right) dx.\end{aligned}$$

Suppose that $\frac{dx}{dt}$ is such that $\frac{d}{dt}\int_{x_{i-1}(t)}^{x_i(t)} u dx = 0.$

Then

$$\int_{x_{i-1}(t)}^{x_i(t)} \frac{\partial}{\partial x}\left(u\frac{\partial}{\partial x}\left[\frac{u^4}{4} + \phi\right]\right) dx = 0. \tag{6.1}$$

A solution of (6.1) for which $\phi = 0$ when $u = 0$ which is

$$\frac{u^4}{4} + \phi = 0.$$

Taking $\phi = -\frac{u^4}{4}$ and $\frac{dx}{dt} = \frac{\partial \phi}{\partial x}$, gives

$$\frac{dx}{dt} = -u^3 \frac{\partial u}{\partial x}$$

everywhere. This equation tells us how the grid points move such that the area under the graph of u between each pair of points remains constant for all time.

6.2 Implementing this Method

6.2.1 Integrating using Fourth Order Runge-Kutta [14]

Take an initial condition and a grid equally spaced over the domain of this initial condition. Calculate the inital areas under the graph of u between grid points using the trapezium rule,

$$A_i = \frac{1}{2}\left(x_i - x_{i-1}\right)\left(u(x_i) + u(x_{i-1})\right).$$

These areas should remain fixed,approximately, for the rest of the problem. Integrate $\frac{dx}{dt} = -u^3 \frac{\partial u}{\partial x} = f(x)$ using fourth order Runge-Kutta.

$$\begin{aligned}
x_i^{n+1} - x_i^n &= \frac{\Delta t}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right) \\
K_1 &= f\left(x_i^n\right) \\
K_2 &= f\left(x_i^n + \frac{\Delta t}{2} K_1\right) \\
K_3 &= f\left(x_i^n + \frac{\Delta t}{2} K_2\right) \\
K_4 &= f\left(x_i^n + \Delta t K_3\right)
\end{aligned}$$

The derivative $\frac{\partial u}{\partial x}$ required in the function f can be calculated analytically provided the initial condition is differentiable. The solution given by the Runge-Kutta method gives the new grid points. The solution at these grid points can be constructed using the trapezium rule since the area between grid points has remained unchanged. The solution at the boundaries is known to be zero since the grid points are moving such that they always exactly cover the whole domain of the solution. Hence we have

$$u_i = \frac{2A_i}{x_i - x_{i-1}} - u_{i-1}. \tag{6.2}$$

This process must now be repeated each time step until the end time has been reached. However, we now do not have an analytic function for u and hence cannot find $\frac{\partial u}{\partial x}$ analytically. Instead, we use an upwind finite difference, given by

$$\left.\frac{\partial u}{\partial x}\right|_i = \frac{u_i - u_{i-1}}{x_i - x_{i-1}} \qquad x_i > 0$$

$$\left.\frac{\partial u}{\partial x}\right|_i = \frac{u_{i+1} - u_i}{x_{i+1} - x_i} \qquad x_i < 0. \tag{6.3}$$

The differential equation $\frac{dx}{dt} = -u^3\frac{\partial u}{\partial x} = f(x)$ is then solved as before using fourth order Runge-Kutta. Where the function in the Runge-Kutta method needs to be evaluated at x values that are not exact grid points (for example, $x_i^n + \frac{\Delta t}{2}K_{1(i)}$ will not be on a grid point) then the value of u at this point should be found using linear interpolation on the values of u at the grid points on either side of this point. The value of $\frac{\partial u}{\partial x}$ should be found numerically as given in equation (6.3) but using $u(x_i^n + \frac{\Delta t}{2}K_{1(i)})$ and $u(x_{i-1}^n + \frac{\Delta t}{2}K_{1(i-1)})$ in the case where $x_i^n + \frac{\Delta t}{2}K_{1(i)} > 0$ and $u(x_i^n + \frac{\Delta t}{2}K_{1(i)})$

and $u(x_{i+1}^n + \frac{\Delta t}{2} K_{1(i+1)})$ in the case where $x_i^n + \frac{\Delta t}{2} K_{1(i)} < 0$. The values of u at the grid points can then be constructed by the trapezium rule (see (6.2)) and the process is repeated until end time is reached.

This Runge-Kutta method is stable when $\Delta t|\lambda| < 2.78$ where λ is the largest eigenvalue of the Jacobian matrix. Since the Jacobian, $J = \left\{ \frac{\partial f_i}{\partial x_j} \right\}$ depends on the cell size, the method will have a smaller region of stability for a smaller initial cell size, in general.

6.2.2 Integrating using Second Order Backward Differentiation

Instead of using the fourth order Runge-Kutta method with a stability barrier as explained above, the differential equation can be solved using a second order backward difference method. This method is stable for all values of Δt. The scheme is specified as

$$x_i^{n+1} - \frac{4}{3} x_i^n + \frac{1}{3} x_i^{n-1} = \frac{2}{3} \Delta t f(x^{n+1}).$$

Since this scheme uses values from two previous time steps to calculate the current value, it is necessary to use a different scheme initially to find x^1 and u^1 (x^0 and u^0 are already known, they are specified in the initial conditions). The second order Runge-Kutta scheme specified in equation (6.4) is used to find x^1 for use in the second order backward differentiation scheme.

$$\begin{aligned} x^{n+1} - x^n &= \frac{\Delta t}{2} (K_1 + K_2) \\ K_1 &= f(x^n) \\ K_2 &= f(x^n + \Delta t K_1) \end{aligned} \tag{6.4}$$

x_i^1 is found for all values of i using this method and the values of u at these grid points are constructed using the trapezium method as before. Both x^0

and x^1 are now known and the backward differentiation scheme can be used to find the values of x_i at all following time steps.

Since $f(x_i^{n+1})$ is not known, the scheme must be iterated as

$$(x_i^{n+1})^{p+1} - \frac{4}{3}x_i^n + \frac{1}{3}x_i^{n-1} = \frac{2}{3}\Delta t f((x_i^{n+1})^p).$$

Here, p is the iteration count and $(x^{n+1})^0$ is taken as x^n. The iteration is repeated until $|(x^{n+1})^{p+1} - (x^{n+1})^p|$ is within a specified range and we have convergence.

6.3 Results

6.3.1 Evolution of Various Initial Conditions

Figure 11 shows the evolution of the solution with non-self-similar initial data

$$u = \begin{cases} \cos\left(\frac{\pi x}{2}\right) & \text{for } |x| < 1 \\ 0 & \text{for } |x| \geqslant 1 \end{cases}$$

using the fourth order Runge Kutta method to do the integration.

Figure 12 shows the evolution of the solution with self-similar initial data

$$u = \begin{cases} \left(1 - \frac{x^2}{3}\right)^{\frac{1}{4}} & \text{for } x^2 < 3 \\ 0 & \text{for } x^2 \geqslant 3 \end{cases}$$

using the second order backward differentiation method to do the integration.

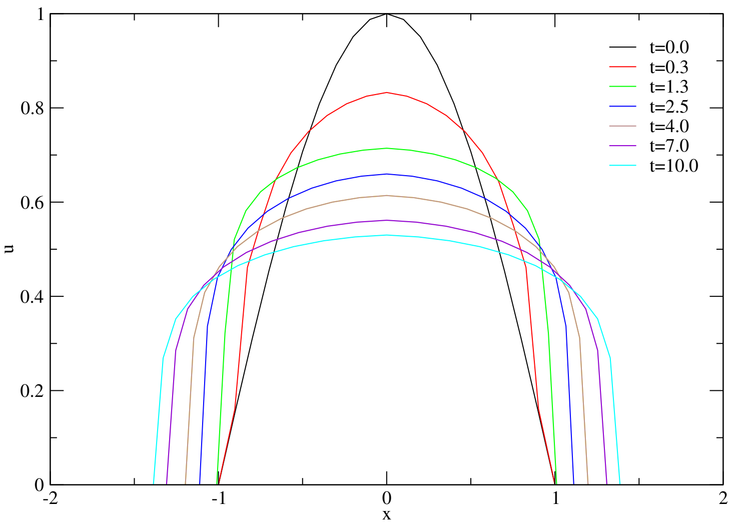

Figure 11: Evolution of solution from scheme using RK4 with initial condition $u = \cos\left(\frac{\pi x}{2}\right)$

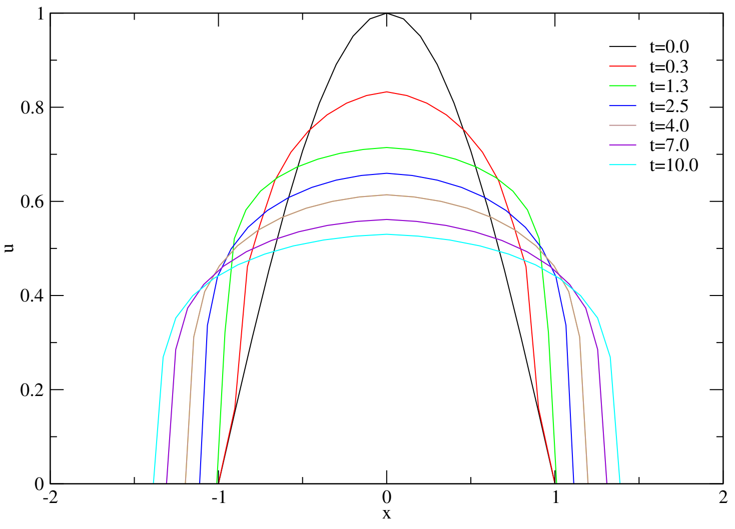

Figure 12: Evolution of solution from scheme using BDF2 with initial condition $u = \left(1 - \frac{x^2}{3}\right)^{\frac{1}{4}}$

6.3.2 The Self-Similar Solution as an Attractor for the Numerical Solution

As with the static mesh methods, the numerical solution should hopefully be sandwiched between two self-similar solutions. The same two initital conditions will be used to test this. The graphs in figures 13 and 14 show that the numerical solutions for both initital conditions are sandwiched between two self-similar solutions for all time.

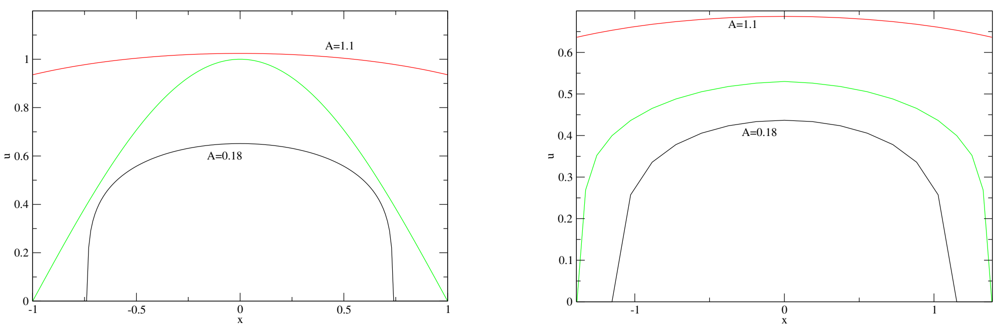

Figure 13: Initial Condition $u = \cos\left(\frac{\pi x}{2}\right)$ sandwiched by two self-similar solutions. Results from RK4 method of integration.

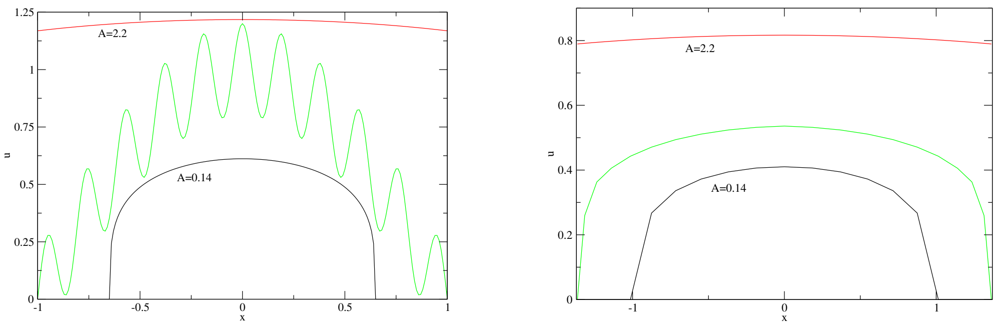

Figure 14: Initial Condition $u = \cos\left(\frac{\pi x}{2}\right) + 0.2\cos\left(\frac{21\pi x}{2}\right)$ sandwiched by two self-similar solutions. Results from RK4 method of integration.

6.3.3 Accuracy of the Numerical Solution

By using the analytic solution at time $t = 1$, the numerical evolution of this solution can be compared to the analytic evolution to find a percentage error in the numerical solution at final time. The initial condition is $u = 1 - \left(\frac{x^2}{3}\right)^{\frac{1}{4}}$. The percentage error will be found by

$$\text{percentage error} = 100 \times \left(\frac{(\sum_i u(i)^2)^{\frac{1}{2}} - (\sum_i ua(i)^2)^{\frac{1}{2}}}{(\sum_i ua(i)^2)^{\frac{1}{2}}} \right).$$

The percentage errors from the different methods are given in the table below.

Method			Percentage Error	Run Time
Conservation Form	Explicit u^4		0.0814%	6.02s
	Implicit u^4	Picard	0.0618%	7.67s
		Newton	0.0625%	7.06s
Non-Conservation	Central Difference		−9.2949%	5.93s
Form	Upwind Difference		−1.9724%	6.13s
Moving Mesh	RK4		0.6592%	5.99s
	BDF2		4.2948%	6.72s

These results show that the implicit methods have the smallest error. They do however have a longer run time. Notice that using Newton-Raphson iteration is quicker than using Picard iteration. This is because Newton-Raphson converges quadratically. The central difference method has a very large error. In the problem used here, the data is advecting in one direction only and in finding the central difference, it is taking data from both directions, causing the numerical solution to depend on data on which the physical solution does not depend.

6.3.4 Conservation of the Integral

If the method used is conservative then it is expected that the integral of the solution will remain constant. It will probably not be equal to the analytic integral due to the discretisation in space. Using the initial condition $u = \cos\left(\frac{\pi x}{2}\right)$, the analytic integral is $\frac{2}{\pi}\left[\sin\left(\frac{\pi x}{2}\right)\right]_{-1}^{1} = \frac{4}{\pi} = 1.27324$.

Method			Time	Integral
Analytic				1.27324
Conservation Form	Explicit u^4		$t=0$	1.27321
			$t=10$	1.27321
	Implicit u^4	Picard	$t=0$	1.27321
			$t=10$	1.27321
		Newton	$t=0$	1.27321
			$t=10$	1.27321
Non-Conservation	Central Difference		$t=0$	1.27321
Form			$t=10$	1.05728
	Upwind Difference		$t=0$	1.27321
			$t=10$	1.23091
Moving Mesh	RK4		$t=0$	1.27321
			$t=10$	1.27321
	BDF2		$t=0$	1.27321
			$t=10$	1.27321

The static mesh methods all have $\Delta x = 0.01$ and the moving mesh method has an initial regular mesh with $\Delta x = 0.01$. The moving mesh method is based on the area under the solution curve remaining constant between each grid point so must conserve the integral of the solution. The numerical methods based on the PDE in conservation form are shown to be conservative, as expected.

7 Conclusions

For this dissertation, I have looked mainly at a number of different numerical schemes that can be used to solve the non-linear diffusion equation.

Initially, I investigated scale invariance and how it was used to obtain a family of self-similar solutions to the problem. This family of solutions acted as an attractor for more general solutions.

I have looked at different static mesh methods for solving the equation in both conservation form and non-conservation form. I also looked at a moving mesh method.

In conservation form, I used the Crank-Nicolson scheme with three different solvers. First, the equation was linearised in u^{n+1} and a standard solver used. The fully non-linear equation was then solved using Picard iteration and the Newton-Raphson method.

In non-conservation form, the equation was linearised in u^{n+1} and both upwind and central differences were used in the scheme.

All the static mesh methods have also been applied to the case where the diffusion coefficient is not known analytically but is in the form of a table of values.

For the moving mesh method, the integral of the solution is kept constant between consecutive grid points throughout the entire calculation and this fact is used to form an ordinary differential equation which can be solved to give the new positions of the grid points at the next time step. This differential equation was solved in two different ways, using fourth order Runge-Kutta and using second order backward differentiation with the initial values given by second order Runge-Kutta.

All the methods carried the property of the analytic solution that a general solution should tend to the self-similar solution. Even an oscillatory initial condition remains sandwiched by two self-similar solutions for all methods.
For the static mesh methods where the PDE was in conservation form, we have seen that there is a trade off between the accuracy of the final solution and the run time. The iterative methods ran more slowly but gave a more accurate answer. The Newton-Raphson seems to be the best scheme here since the iterations converge quadratically so it has a significantly shorter run time than the Picard iteration but still maintains good accuracy.
The methods used on the equation in non-conservative form have quite poor accuracy. The central difference especially is very poor due to including data in the numerical derivatives that the solution is not actually dependent on.
The moving mesh method is less accurate than the static mesh conservative methods but has the advantage of only solving on the area of grid where the solution exists. If the problem were run to a much later end time, this method would prove much more efficient since the final grid would be significantly larger than the initial grid and the static mesh methods would have to solve on the whole grid throughout the entire problem.

8 Further Work

The most successful method considered here is the Newton-Raphson method. However, this requires the complete formation of the Jacobian matrix. With more time, I would like to have investigated a group of methods called Newton-Krylov methods that do not require the explicit formation of the Jacobian. I will give a short summary here of how these methods work. To implement a Krylov method, we need only represent the matrix-vector product rather than explicitly represent the whole matrix. This allows the definition of a Jacobian free algorithm with an approximation

$$J(\mathbf{u})\mathbf{v} \approx \frac{F(\mathbf{u}+\epsilon\mathbf{v}) - F(\mathbf{u})}{\epsilon}, \tag{8.1}$$

where F is as given in (3.5), $\mathbf{v}$ is a Krylov vector and $\epsilon = 10^{-8}(1 + ||\mathbf{u}||)$. For the algorithm to be effective, a preconditioner must be used. Right preconditioning by the matrix $\tilde{M}$ is used and we need to approximate $J\tilde{M}^{-1}\mathbf{v}$. This is done in two steps. First, we must solve $M\mathbf{y} = \mathbf{v}$. In [7], the use of a Picard-type iteration with an approximate solution computed with a single multigrid cycle is recommended. We can then approximate the Jacobian via

$$J\tilde{M}^{-1}\mathbf{v} = J\mathbf{y} \approx \frac{F(\mathbf{u}+\epsilon\mathbf{y}) - F(\mathbf{u})}{\epsilon}. \tag{8.2}$$

The overall iteration takes the form $(J\tilde{M}^{-1})(\tilde{M}\delta u) = -F(u)$. The full algorithm to solve the problem can be stated as follows [7]:

1. Start the nonlinear iteration, $k = 0$.

2. Compute the nonlinear residual, $\mathbf{r} = -F(\mathbf{u})$.

   (a) Start the Krylov iteration to solve $J\delta u = \mathbf{r}$, $n = 0$. Initialise the Krylov vector with $\mathbf{v}_n = \mathbf{r}_n$.

(b) Compute the preconditoned Krylov vector, $J\tilde{M}^{-1}\mathbf{v}$, using a multigrid cycle to approximate the solution to $A\mathbf{y}_n = \mathbf{v}_n$.

(c) Perform the matrix-vector multiply through the operation $\mathbf{w}_n = \dfrac{F(\mathbf{u} + \epsilon\mathbf{y}_n) - F(\mathbf{u})}{\epsilon}$.

(d) Complete the Krylov iteration, $\mathbf{v}_{n+1} = \dfrac{\mathbf{w}_n}{||\mathbf{w}_n||_2}$ and compute convergence. If converged, exit, otherwise $n := n + 1$ and go to (b).

3. Compute the update to the full nonlinear problem.

4. Check for nonlinear convergence. If converged, exit; otherwise, $k := k + 1$ and go to 2.

This method proves to be more accurate than the standard Crank-Nicolson method with Picard iteration. Figure 15 from [7] shows the propogation of a one dimensional radiation heat wave (also referred to as a Marshak wave [15]). The system was run using four different solvers, backward Euler with Picard iteration, Crank-Nicolson with Picard iteration, backward Euler with Newton-Krylov iteration and Crank-Nicolson with Newton-Krylov iteration. It is obvious from this graph that the solutions using Newton-Krylov iteration are much closer to the exact solution than those using Picard iteration.

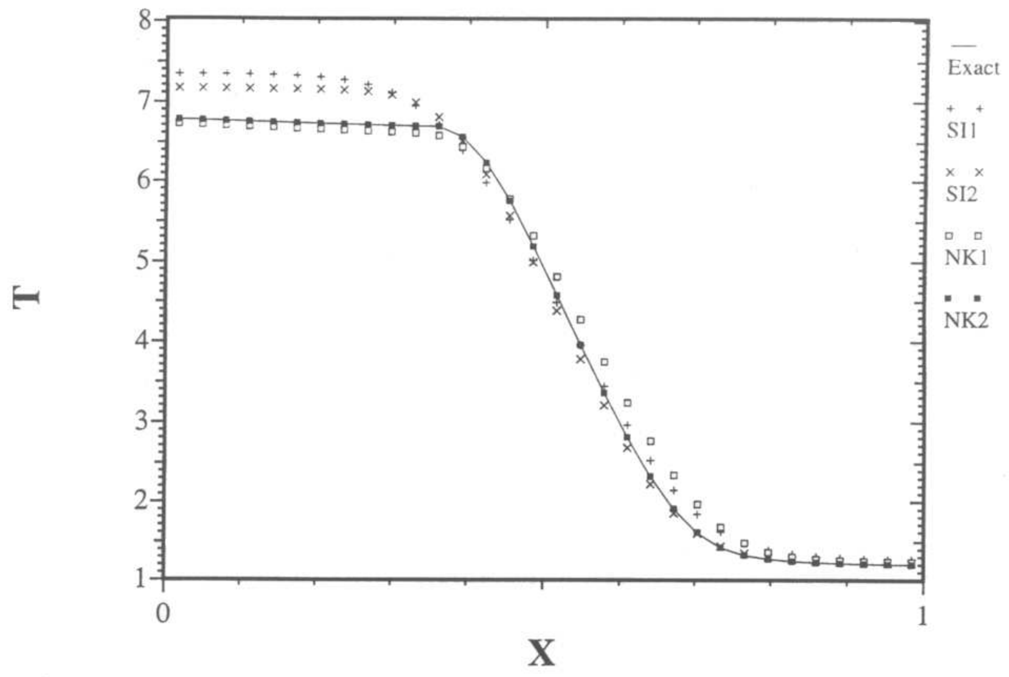

Figure 15: The one-dimensional Marshak wave problem used to demonstrate the accuracy of various non-linear iterative techniques.

References

[1] D.J.Tritton. *Physical Fluid Dynamics.* Oxford Science Publications, 1988.

[2] K.W.Morton & D.F.Mayers. *Numerical Solution of Partial Differential Equations.* Cambridge University Press, 1994.

[3] J.D.Murray. *Mathematical Biology.* Springer Verlag, 1989.

[4] C.J.Budd & M.D.Piggott. *Geometric Integration and its Applications*, Handbook of Numerical Analysis Vol XI (2003), 35-135.

[5] C.J.Budd, G.J.Collins, W.Huang & R.D.Russell. *Self-Similar Numerical Solutions of the Porous-Medium Equation using Moving Mesh Methods*, Phil. Trans. R. Soc. Lond. A (1999) 357, 1047-1077.

[6] P.J.Capon. *Adaptive Stable Finite Element Methods for the Compressible Navier-Stokes Equation.* PhD Thesis, University of Leeds, 1995.

[7] W.J.Rider, D.A.Knoll & G.L.Olson. *A Mulitgrid Newton-Krylov Method for Multimaterial Equilibrium Radiation Diffusion*, Journal of Computational Physics (1999) 152, 164-191.

[8] G.I.Barenblatt. *Scaling, Self-Similarity and Intermediate Asymptotics.* Cambridge University Press. 1996.

[9] http://mathworld.wolfram.com/LeastSquaresFitting.html

[10] E.S.Oran & J.P.Boris. *Numerical Simulation of Reactive Flow.* Elsevier, 1987.

[11] http://physchem.ox.ac.uk/~rgc/john/Thesis/2/2.html

[12] B.V.Wells. *A Moving Mesh Finite Element Method for the Numerical Solution of Partial Differential Equations and Systems.* PhD Thesis, Univeristy of Reading, 2004.

[13] Y.Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Co., Boston, 1996.

Available online at: http://www-users.cs.umn.edu/~saad/books.html

[14] K.W.Blake. *Moving Mesh Methods for Nonlinear Parabolic Partial Differential Equations.* PhD Thesis, University of Reading, 2001.

[15] S.L.Lee, C.S.Woodward, F.Graziani. *Analyzing Radiation Diffusion using Time-Dependent Sensitivity-Based Techniques.* March 13 2003.

Available online at: http://www.llnl.gov/CASC/nsde/pubs/diffusion_sens.pdf