ADAPTIVE SPACE-MESHING STRATEGIES FOR THE NUMERICAL SOLUTION OF PARABOLIC PARTIAL DIFFERENTIAL EQUATIONS IN ONE SPACE DIMENSION

Paul P. Neary B.Sc.

Submitted in fulfillment of an M.Sc. degree by research.

Dublin City University
Dr. J. Carroll (supervisor)
School of Mathematical Sciences

June 1990

Declaration

I declare that this dissertation is entirely my own work and that it has not been submitted to this or any other University as an exercise for a degree.

Signed ....Paul Neary.

Contents

Acknowledgements

I wish to express my overwhelming gratitude and thanks to my own family, for their immense patience and encouragement during the course of my postgraduate studies.

I would also like to thank my project supervisor at Dublin City University, Dr. John Carroll, for his guidance and advice and my former group leader at Philips Research (Eindhoven), Dr. Paul Hart, for allowing me the use of computing resources at the laboratory.

Lastly, thanks to my colleague, Ad van Run, for the pleasure of sharing many fruitful discussions on subjects related to this thesis.

Paul P. Neary

Adaptive Space-Meshing Strategies for the Numerical Solution of Parabolic Partial Differential Equations in One Space Dimension.

Abstract

The effectiveness of adaptive space-meshing in the solution of one-dimensional parabolic partial differential equations (PDEs) is assessed.

Present day PDE software typically involves discretisation in space (using Finite Differences or Finite Elements) to produce a system of ordinary differential equations (ODEs) which is then solved routinely using currently available high quality ODE integrators. Such approaches do not attempt to control the errors in the spatial discretisation and the task of ensuring an effective spatial approximation and numerical grid are left entirely to the user. Numerical experiments with Burgers' equation demonstrate the inadequacies of this approach and suggest the need for adaptive spatial meshing as the problem evolves. The currently used adaptive meshing techniques for parabolic problems are reviewed and two effective strategies are selected for study. Numerical experiments demonstrate their effectiveness in terms of reduced computational overhead and increased accuracy. From these experiences possible future trends in adaptive meshing can be identified.

Keywords: Partial Differential Equations, Parabolic problems, Method of Lines, Adaptive meshing, Finite Difference Methods, Finite Element Methods, Moving Finite Elements, Lagrangian schemes, Linearly implicit ODE systems.

Chapter 1

Introduction

Partial differential equations (PDEs) occur widely in Science and Engineering in the modelling of continuum problems. This thesis is concerned with the automatic solution of parabolic PDEs in one space dimension and the effectiveness of incorporating adaptive space-meshing algorithms into existing software.

Chapter 2 introduces scalar and vector systems of parabolic PDEs and outlines the general nature of parabolic problems. With reference to suitable examples the need for adaptive space-meshing is established.

Historically the development of PDE software has been slow owing to the great diversity of such problems. This is in sharp contrast to the now highly developed software for ordinary differential equations (ODEs). Consequently the most successful PDE software, for example PDEONE [46] and PDECOL [33], have involved the discretisation of the problem in the space dimension (semidiscretisation) and its reduction to a system of ODEs which can be routinely solved using available high quality ODE integrators. When semidiscretisation is performed using Finite Differences the procedure is known as the "method of lines". Alternatively Finite Element methods (Galerkin [18] and Collocation [38]) may also be used to perform the semidiscretisation.

Chapter 3 begins by outlining the presently used methods of semidiscretisation. The nature of the resulting ODE system is discussed and this leads to the notion of *stiffness*. The semidiscretisation of parabolic problems generally results in systems of ODEs which are at least "mildly" stiff. The presently used methods of integration for stiff systems of ODEs are outlined and the standard packages reviewed.

Current ODE software performs the temporal integration more or less automatically and the techniques for controlling the integration (some of which are heuristic) are outlined. The combination of semidiscretisation followed by ODE integration are the hallmarks of present day PDE solvers. Two standard PDE packages are reviewed. In these packages no error control for the spatial discretisation is attempted and spatial errors are presumed to be negligible. Some numerical results for Burgers' equation demonstrate the inadequacies of this approach for problems involving propagating shocks and/or boundary layers. In such cases the solutions are computationally expensive to obtain and are often inaccurate. The need to adapt the mesh as the problem evolves becomes clear.

The inadequacy of uniform spatial grids in the solution of parabolic equations may be explained by analysing the typical methods of semidiscretisation used for such problems. The use of non-uniform spatial meshes, however, offers the possibility of much improved results over the uniform mesh approach. This is analogous to the existing use of non-uniform grids in the time discretisation of such equations, and in the solution of general ODEs. The ultimate aim in both situations is to evenly distribute the numerical errors over the problem domain. The methods of determining suitable non-uniform grids for particular problems (numerical grid generation) are considered. Fixed non-uniform grids are, however, of limited use when solving parabolic type problems because the spatial structure of the solution typically alters with time. In order to ensure adequate spatial resolution throughout the problem evolution, regeneration of the non-uniform grid is necessary. This is the principle underlying the adaptive mesh approach. Adaptive mesh strategies involve a particular grid generation method and a dynamic coupling between the grid and the evolving solution. Current trends in adaptive meshing are discussed and the philosophies behind the various approaches are contrasted. Efficiency, robustness and versatility are the characteristic features of superior adaptive mesh strategies.

In Chapter 6 the algorithms selected in the previous chapter are implemented in conjunction with a standard ODE solver. Comparisons are made between uniform and adaptive mesh implementations for several example problems.

Chapter 7 concludes with a summary of findings regarding the automatic solution of Parabolic PDEs. The relative merits of adaptive and non-adaptive strategies are discussed particularly in relation to computational expense and efficiency. Finally areas for future research are identified.

Chapter 2

Parabolic Partial Differential Equations

2.1 Classification of partial differential equations

We are interested in the solution of linear and nonlinear parabolic equations and systems of equations. Firstly by considering the linear case of the second order partial differential equation and establishing what form of auxiliary conditions serve to determine a unique solution a three-way classification of such equations is possible.

The general form of a linear 2nd order PDE in two independent variables x,t is

$$\begin{aligned} A(x,t)U_{xx} + 2B(x,t)U_{xt} + C(x,t)U_{tt} \quad & + \\ D(x,t)U_x + E(x,t)U_t + F(x,t)U + G(x,t) \quad & = 0 \end{aligned} \tag{2.1}$$

Subscripts imply partial derivatives and the coefficient 2B is chosen for later convenience.

A solution of (2.1) in a region $\mathcal{R}$ of the (x,t) plane is a function U(x,t) for which U and the partial derivatives U_x and U_t are defined at each point (x,t) in $\mathcal{R}$ and for which the equation reduces to an identity at each such point.

To determine the form of auxiliary data required to guarantee a unique solution to (2.1) consider the specification of U along some interval of the y axis, U(0,t)=f(t). This allows calculation of all partial derivatives of U w.r.t. t along the interval (0,t) presuming that f(t) is sufficiently differentiable. No information about the partial derivatives w.r.t. x is known except that (2.1) relates U_{xx} to U_x. Therefore prescribing $U_x(0,t) = g(t)$ (the normal derivative) along the chosen interval allows the calculation of U_{xt} and further derivatives w.r.t. U_{xx} can now be determined from (2.1).

It is now possible to construct a Taylor series representation of U in the neighbourhood of (0,t). This suggests that a unique solution to (2.1) is possible given the function U and its normal derivative along one axis where both the functions and coefficients appearing in (2.1) are sufficiently differentiable.

The formal statement of the above procedure is the Cauchy-Kowalewski Theorem which states that if f(t), g(t) and the set of coefficients

$$\frac{B(x,t)}{A(x,t)}, \frac{C(x,t)}{A(x,t)} \cdots \frac{G(x,t)}{A(x,t)}$$

are analytic in the neighbourhood of some point $(0, t_0)$ then the above procedure will generate a unique solution to (2.1) which is also analytic in the neighbourhood of $(0, t_0)$. We call the data f(t) and g(t) the *Cauchy data* for the problem (2.1) and such a problem is called a *Cauchy problem.*

By a further generalisation one may seek the solution of (2.1) given Cauchy data specified along an arbitrary curve Γ in $\mathcal{R}$. Some equations of the form (2.1) possess solutions for all choices of Γ whereas others require a restriction on the choice of Γ. The various cases allow the characterisation of partial differential equations into three distinct classes. Each type of equation has particular properties concerning the dependency of solutions upon the auxiliary data.

Consider again the Cauchy problem where the Cauchy data is specified along an arbitrary curve Γ. The 1^{st} order partial derivatives are known.

$$p = \frac{\partial U}{\partial x} \quad and \quad q = \frac{\partial U}{\partial t}$$

The 2^{nd} order derivatives are also required. They are denoted by

$$r = \frac{\partial^2 U}{\partial x^2}, \quad s = \frac{\partial^2 U}{\partial t^2} \quad and \quad u = \frac{\partial^2 U}{\partial x \partial t}$$

Three relationships between r,s and t are known. These are given by equation (2.1) and the total derivatives of p and q as follows.

$$\begin{aligned}
Ar + 2Bs + Cu &= \phi(x, t, U, p, q) \\
dp &= \frac{\partial p}{\partial x} dx + \frac{\partial p}{\partial t} dt \\
&= r.dx + s.dy \\
dq &= \frac{\partial q}{\partial x} dx + \frac{\partial q}{\partial t} dt \\
&= s.dx + u.dt
\end{aligned}$$

or

$$\begin{bmatrix} A & 2B & C \\ dx & dt & 0 \\ 0 & dx & dt \end{bmatrix} \begin{bmatrix} r \\ s \\ u \end{bmatrix} = \begin{bmatrix} \phi \\ dp \\ dq \end{bmatrix} \tag{2.2}$$

A unique solution to this system of equations is possible if its determinant is non-zero. Conversely if the determinant is zero then the second order derivatives are indeterminate and the original partial differential equation (2.1) has no solution. The case of zero determinant yields the following equation.

$$A.dt^2 + 2B.dx.dt + C.dx^2 = 0 \tag{2.3}$$

The solutions of this quadratic equation determine two families of curves along which the specification of Cauchy data is insufficient to guarantee a unique solution to (2.1). These curves are called **characteristics**. The solutions are

$$dy = \frac{-B \pm \sqrt{B^2 - AC}}{A} \tag{2.4}$$

Equations of the form (2.1) are characterised as being **hyperbolic**, **parabolic** or **elliptic** depending on the sign of the discriminant $B^2 - AC$.[1]

Hyperbolic case $B^2 - AC > 0$: Here there exists two real families of characteristics. The prototype hyperbolic equation is the *wave* equation.

$$\frac{\partial^2 U}{\partial x^2} - \frac{\partial^2 U}{\partial t^2} = 0$$

Hyperbolic problems represent the propagation of signals with finite speed. The trajectories of these signal fronts correspond to the characteristic curves.

Parabolic case $B^2 - AC = 0$: Here the twin families of characteristics have degenerated into one family. The prototype parabolic equation is the *heat* equation.

$$\frac{\partial^2 U}{\partial x^2} - \frac{\partial U}{\partial y} = 0$$

Parabolic problems represent the propagation of signals with infinite speed.

[1]These names arise due to the similarity of equation (2.1) with the quadratic form $\alpha x^2 + 2\beta xt + \gamma t^2 + \delta x + \epsilon t = const.$ which gives rise to the hyperbola, parabola and ellipse under the same conditions.

Elliptic case $B^2 - AC < 0$: Here there are no real characteristics since the solutions of the quadratic are imaginary. Thus there is no restriction on the choice of curve Γ along which Cauchy data may be specified. The prototype elliptic equation is the *Laplace* equation.

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial t^2} = 0$$

Although the preceding classification would appear to assign a set type to every partial differential equation it does however depend on the region of the (x,t) plane under consideration. For example the equation

$$x\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial t^2} = 0$$

is elliptic in the region $x > 0$, hyperbolic in the region $x < 0$ and parabolic for $x = 0$. Subsequently, in section 2.2, it will be seen that solutions exhibit interesting features in regions where a transition between different types of equations occur.

2.2 The nature of parabolic problems

Although the Cauchy-Kowalewski Theorem provides a convenient method of characterising partial differential equations it does not however guarantee physically meaningful solutions. In order to guarantee meaningful solutions a problem must be **well-posed.** A problem is well-posed if

- The solution exists.
- The solution is unique.
- The solution depends continuously on the auxiliary data.

The last requirement is reasonable since without it the comparison of theory and experiment would be impossible. Only the Cauchy problem for the hyperbolic equation is well-posed. For elliptic and parabolic equations alternative auxiliary conditions lead to well posed problems.

Let us now focus attention on parabolic problems which are well-posed and examine their properties. Let us consider a general system of $\mathcal{N}$ parabolic equations of the form

$$\begin{aligned}\frac{\partial u_i}{\partial t} &= \frac{\partial}{\partial x}\{g_i(x,t,U_i)\frac{\partial U_i}{\partial x}\} + f_i\{x,t,\vec{U},\frac{\partial U_i}{\partial x}\} \\ \vec{U} &= [U_1,U_2,\ldots]^T \quad i = 1,2\ldots\mathcal{N}. \qquad (2.5)\end{aligned}$$

This includes a wide variety of both linear and nonlinear equations.

The most commonly posed problem involving one-dimensional parabolic partial differential equations is the solution of (2.5) in the semi-infinite strip $a \leq x \leq b, \quad t \geq 0$ subject to the initial and boundary conditions

$$\begin{aligned}U_i &= f_i(x) \quad t = 0 \\ p_i(t)U_i + q_i(t)\frac{\partial U_i}{\partial x} &= r(U_i,t) \quad x = a,b \quad t > 0 \\ & i = 1,2\ldots\mathcal{N}. \qquad (2.6)\end{aligned}$$

The stated boundary conditions are general but may degenerate to Dirichlet and Neumann conditions. The independent variables t and x typically represent time and space-like quantities. In most practical examples the order $\mathcal{N}$ of the system of equations does not exceed 3 or 4.

Below are some examples of both linear and nonlinear parabolic equations and systems of equations. These examples were chosen to demonstrate the diversity of parabolic problems and serve as a suite of test problems for the numerical methods to be discussed in subsequent chapters.

2.2.1 Linear problems

Consider the simple heat equations of problems 1 and 2. These equations

Problem 1

$$\begin{aligned}
\frac{\partial U}{\partial t} &= \frac{1}{\pi^2}\frac{\partial^2 U}{\partial x^2} \qquad 0 \le x \le 1 \\
U(0,t) &= U(1,t) = 0 \\
U(x,0) &= \sin(\pi x) \\
U(x,t) &= e^{-t}\sin(\pi x)
\end{aligned}$$

Source: Davis and Flaherty. [14]

Problem 2

$$\begin{aligned}
\frac{\partial U}{\partial t} &= \frac{\partial^2 U}{\partial x^2} \qquad 0 \le x \le 1 \\
\frac{\partial U}{\partial x}(0,t) &= \pi^2 e^{-\pi^2 t} \qquad \frac{\partial U}{\partial x}(1,t) = -\pi^2 e^{-\pi^2 t} \\
U(x,0) &= \sin(\pi x) \\
U(x,t) &= e^{-\pi^2 t}\sin(\pi x)
\end{aligned}$$

Source: Mitchell and Griffiths. [37]

constitute the simplest parabolic problems. Physically they often represent the diffusion of heat within a rod of constant cross-section which is insulated along its length. The coefficient multiplying $\frac{\partial^2 U}{\partial x^2}$ is called the diffusivity. Both equations are easily solved using the method of ***separation of variables.***

Numerically the solution of these equations is fairly routine except in the case of Problem 2 where the presence of Neumann boundary conditions complicates the solution procedure somewhat. The accurate representation of boundary conditions is one of the main problems encountered in the numerical solution of partial differential equations. In Chapter 3 it will be seen how the use of higher accuracy spatial discretisations is restricted by such difficulties.

Problems 3 and 4 are slightly more complex in that non-derivative terms appear. Such terms are called ***sources*** or ***sinks*** depending on whether their signs are positive or negative. In heat conduction problems they represent internal sources or losses of heat. Unlike the previous problems the steady state solution $(U(x, t \to \infty))$ is not zero but some function of x. Problem 3 has a small parameter ϵ multiplying the time derivative. The magnitude of this parameter determines how quickly the problem reaches steady state. For very small values of ϵ the steady state is achieved quickly following an initial period of rapid transition. Problem 4 has a parameter σ multiplying the diffusive term $\frac{\partial^2 U}{\partial x^2}$. Davis and Flaherty used this problem to test the performance of an adaptive grid. The solution comprises a travelling wave whose speed depends on the values r_1 and r_2. For small σ one would expect the term $\frac{\partial^2 U}{\partial x^2}$ to have little effect on the solution and for the wave to maintain its amplitude during its propagation.

The third and perhaps the most common type of parabolic equation is the **convection-diffusion** equation. Problems 5 and 6 are examples of convection-diffusion equations. These equations represent the interplay between convection and diffusion. Of importance here is the relative magnitude of the convection and diffusion coefficients. For instance consider the steady state of problem 5. The time dependent term vanishes leaving the equation

$$\epsilon \frac{\partial^2 U}{\partial x^2} - k \frac{\partial U}{\partial x} = 0$$

Problem 3

$$\begin{aligned}
\epsilon\frac{\partial U}{\partial t} &= \frac{\partial^2 U}{\partial x^2} + 1 - 3x \qquad 0 \le x \le 1 \\
U(0,t) &= U(1,t) = 0 \\
U(x,0) &= \sin(\pi x) + \frac{x^2}{2}(x-1) \\
U(x,t) &= e^{-\frac{\pi^2 t}{\epsilon}}\sin(\pi x) + \frac{x^2}{2}(x-1)
\end{aligned}$$

Source: Philips and Rose. [44]($\epsilon = 10^{-2}$).

Problem 4

$$\begin{aligned}
\frac{\partial U}{\partial t} &= \sigma\frac{\partial^2 U}{\partial x^2} + f(x) \qquad 0 \le x \le 1 \\
U(0,t) &= \tanh(r_2 t - r_1) \qquad U(1,t) = \tanh(r_2 t) \\
U(x,0) &= \tanh(r_1(x-1)) \\
U(x,t) &= \tanh(r_1(x-1) + r_2 t)
\end{aligned}$$

f(x) is chosen so that the auxiliary data satisfy the equation.
Source: Davis and Flaherty. [14] ($\sigma = 10^{-2}, r_1 = r_2 = 5$).

Problem 5

$$\begin{aligned}
\epsilon\frac{\partial U}{\partial t} &= \epsilon\frac{\partial^2 U}{\partial x^2} - k\frac{\partial U}{\partial x} \qquad 0 \le x \le 1 \\
U(0,t) &= 0 \qquad U(1,t) = 1 \\
U(x,0) &= 0 \\
U(x,t) &= \frac{e^{kx/\epsilon} - 1}{e^{k/\epsilon} - 1} \sum_{n=1}^{\infty} \frac{(-1)^n n\pi}{(n\pi)^2 + \frac{k}{2\epsilon}} e^{\frac{k(x-1)}{2\epsilon}} \sin(n\pi x) e^{-[(n\pi)^2\epsilon + \frac{k^2}{4\epsilon}]t}
\end{aligned}$$

Source: Evans and Abdullah.[17]

Problem 6

$$\begin{aligned}
\epsilon\frac{\partial U}{\partial t} &= a(x,t)\frac{\partial^2 U}{\partial x^2} + b(x,t)\frac{\partial U}{\partial x} \qquad 0 \le x \le 1 \\
U(0,t) &= e^{t+2} \qquad U(1,t) = e^{2(t+2} \\
U(x,0) &= e^{2(x+1)} \\
U(x,t) &= e^{(x+1)(t+2)} \\
a(x,t) &= \frac{(x+1)}{2(t+2)^2} \qquad b(x,t) = \frac{(x+1)}{2(t+2)}
\end{aligned}$$

Source: Ciment et. al.[9]

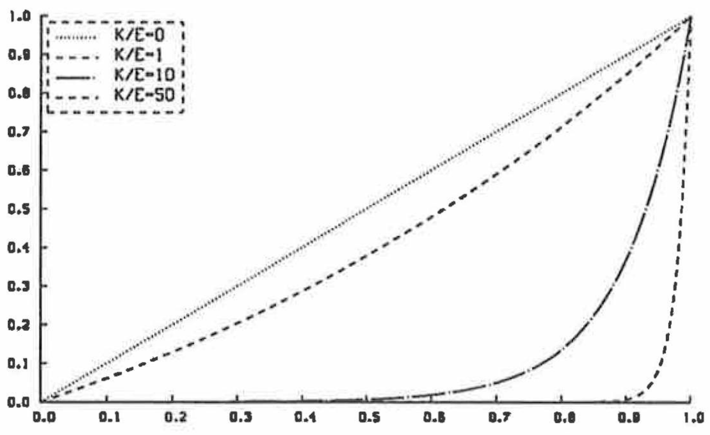


Figure 2.1: Exact solution of the steady state convection-diffusion equation.

subject to the boundary conditions $U(0) = 0$ and $U(1) = 1$. The exact solution to this problem is

$$U = \frac{e^{kx/\epsilon} - 1}{e^{k/\epsilon} - 1} \tag{2.7}$$

Figure 2.1 shows the solution for successively larger values of k/ϵ. For large k/ϵ values the solution is almost unchanged until close to the downstream boundary at x=1. A rapid change in U occurs at x=1 in order to accommodate the boundary condition. For most of the interval $0 \leq x \leq 1$ the problem is *convection-dominated.* However, in the thin layer adjacent to x=1 the slope of U is changing so rapidly that the diffusive term is no longer insignificant. This phenomenon is termed a *boundary layer.* Within this layer the problem is parabolic in nature whereas elsewhere it is very nearly hyperbolic. Problem 5 exhibits a boundary layer at x=1 for large values of k/ϵ. Problems involving boundary layers cause computational difficulties which will be studied in Chapter 3 to follow.

Problem 6 is a convection-diffusion problem in which the coefficients are variable. At t=0 the coefficient b is twice the magnitude of coefficient a and, as t increases, their relative magnitudes become more disparate. Eventually the problem becomes convection-dominated and a boundary layer forms at x=1.

2.2.2 Nonlinear problems

In the study of nonlinear parabolic equations the most popular and widely studied equation is Burgers' equation.

Problem 7

$$\frac{\partial U}{\partial t} = \epsilon\frac{\partial^2 U}{\partial x^2} - U\frac{\partial U}{\partial x} \qquad 0 \le x \le 1$$

This equation is one of the few authentically nonlinear equations for which exact solutions are available. Burgers' equation is a very good model for the Navier-Stokes equations since it represents in the simplest manner possible the balance between the nonlinear convective process $(U\frac{\partial U}{\partial x})$ and the dissipative process $(\epsilon\frac{\partial^2 U}{\partial x^2})$. Burgers' equation is closely related to the ***kinematic wave equation***

$$\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} = 0$$

This hyperbolic equation exhibits discontinuous solutions. For example, if the initial conditions are $U = \exp^{-x^2}$ then discontinuities appear for $T > \sqrt{\exp/2}$. In the case of Burgers' equation however, the term $(\epsilon\frac{\partial^2 U}{\partial x^2})$ prevents the solution from becoming multivalued and for small $(\epsilon\frac{\partial^2 U}{\partial x^2})$ a boundary layer develops. This type of behaviour is typical of "nearly" hyperbolic equations where the developing shocks are "smeared" out by the diffusive term.

Two solutions for Burgers' equation are given in (a) and (b) below. For the

(a) propagating sine wave solution

$$\begin{aligned}
U(0,t) &= U(1,t) = 0 \\
U(x,0) &= \sin(\pi x) \\
U(x,t) &= \frac{4\epsilon\pi \sum_{n=1}^{\infty} \exp(-\epsilon n^2\pi^2 t) n I_n(\frac{U(x,0)}{2\pi\epsilon}) \sin(n\pi x)}{I_0(\frac{U(x,0)}{2\pi\epsilon}) + 2\sum_{n=1}^{\infty} \exp(-\epsilon^2 n^2\pi^2 t) I_n(\frac{U(x,0)}{2\pi\epsilon}) cos(n\pi x)}
\end{aligned}$$

where

$$\begin{aligned}
I_0(z) &= \int_0^1 \exp(z\cos(\pi x))dx \\
I_n(z) &= \int_0^1 \exp(z\cos(\pi x))\cos(n\pi x)dx
\end{aligned}$$

(b) overtaking shocks solution

$$\begin{aligned}
U(0,t) &= U_e(0,t) \qquad U(1,t) = U_e(1,t) \\
U(x,0) &= U_e(x,0) \\
U_e(x,t) &= 1 - 0.9\frac{r_1}{R} - 0.5\frac{r_2}{R} \\
R &= r_1 + r_2 + r_3 \\
r_1 &= e^{\frac{-(x-0.5+4.95t)}{20\epsilon}} \\
r_2 &= e^{\frac{-(x-0.5+0.75t)}{20\epsilon}} \\
r_3 &= e^{\frac{-(x-0.375t)}{2\epsilon}}
\end{aligned} \tag{2.8}$$

sine wave initial condition in (a) the solution was determined by Cole [10]. The solution is a propagating sine wave. As the wave moves downstream (to the right) the nonlinear convective term causes the leading face of the wave to steepen and the diffusive term causes the amplitude of the wave to diminish.

Due to the downstream condition U(1,t)=0 a boundary layer forms at x=1 for small ϵ. This is completely analogous to the convection-diffusion problems discussed earlier. After a time t of $O(\frac{1}{\epsilon})$ the distorted wave is diffused away. The second solution, (b), obtained from Madsen and Sincovec gives rise to overtaking shocks when ϵ is small. Again as t increases a boundary layer develops at x=1 which is of thickness $O(\sqrt{\epsilon})$ and slope $O(\frac{1}{\epsilon})$. In Chapter 6 Burgers' equation will be used as a test problem for the numerical methods to be discussed in Chapters 3 and 4.

Problems 8 and 9 are examples of nonlinear heat equations. In Problem 8 the parameter U^0 can be chosen to control the steepness of the transient solution and k can be used to control the steepness of the steady state solution at x=0. In both problems only analytic expressions for the steady state solutions are available. Problem 9 is nonlinear and also possesses a nonlinear boundary condition at x=1. Problems 10 and 11 consist of two sets of coupled nonlinear parabolic equations. In Problem 10 the equations are coupled through source terms whereas in Problem 11 they are coupled through all spatial terms. In general the greater the coupling between the equations of a system the more difficult is the numerical solution. Problem 12 is a practical problem involving parabolic systems. This problem is taken from the field of semiconductor process modelling. This one-dimensional system represents the diffusion of dopant material in Silicon at high temperatures. The equations are coupled through diffusive and convective terms. The convective terms arise due to electrical interaction of the impurity complexes. The possibility for steep gradients in the solution arises when high concentrations of dopant material are present, resulting in strong electrical interaction.

The diversity of parabolic problems as seen above is a challenge to any software for parabolic equations. In the next chapter we shall see that the method of semidiscretisation enables software to be general enough to cater for such a wide variety of problems.

Problem 8

$$\begin{aligned}
\frac{\partial U}{\partial t} &= \frac{\partial U}{\partial x}\left\{e^{kU}\frac{\partial U}{\partial x}\right\} \qquad 0 \leq x \leq 1 \\
U(0,t) &= U(1,t) = U_0 \\
U(x,0) &= U^0 x \\
U(x,t \to \infty) &= ln(1 + (e^{kU^0} - 1)x)
\end{aligned}$$

Source: Braddock and Noye [4] (U^0, k typically O(1)(7)).

Problem 9

$$\begin{aligned}
\frac{\partial U}{\partial t} &= \frac{\partial U}{\partial x}\left\{\frac{\partial U}{\partial x} - U^2\right\} \qquad 0 \leq x \leq 1 \\
U(0,t) &= 50 \qquad U(1,t) = 1 - \sin(U) \\
U(x,0) &= 100 \\
U(x,t \to \infty) &= 50\sqrt{cosh(rx) - \sinh(rx)c} \\
r &= \sqrt{2} \qquad c = 0.88055353224
\end{aligned}$$

Source: Madsen and Sincovec [33] and NAG D03PAF example problem.

Problem 10

$$\begin{aligned}\frac{\partial U}{\partial t} &= 0.024\frac{\partial^2 U}{\partial x^2} - f(U-V) \qquad 0 \leq x \leq 1\\ \frac{\partial V}{\partial t} &= 0.170\frac{\partial^2 V}{\partial x^2} - f(U-V)\end{aligned}$$

$$\begin{aligned}U_x(0,t) &= 0 \quad U(1,t)=0\\ V(0,t) &= 0 \quad V_x(1,t)=0\\ U(x,0) &= 1 \quad V(x,0)=0\end{aligned}$$

Source: NAG D03PBF example problem.

Problem 11

$$\begin{aligned}\frac{\partial U}{\partial t} &= V^2\frac{\partial^2 U}{\partial x^2} + 2V\frac{\partial U}{\partial x}\frac{\partial V}{\partial x} - UV - U^2 + 10 \qquad 0 \leq x \leq 1\\ \frac{\partial V}{\partial t} &= U2\frac{\partial^2 V}{\partial x^2} + 2U\frac{\partial U}{\partial x}\frac{\partial V}{\partial x} - \frac{\partial^2 U}{\partial x^2} + UV - V^2\end{aligned}$$

$$\begin{aligned}U_x(0,t) &= 1/2 \quad U_x(1,t) = 1/2 - \sin(UV)\\ V(0,t) &= \pi \quad V_x(1,t) = 1 + cos(UV)\\ U(x,0) &= 1 \quad V(x,0)=0\end{aligned}$$

Source: Sincovec and Madsen [33].

Problem 12

$$\begin{aligned}
\frac{\partial C_k}{\partial t} &= \frac{\partial}{\partial x} D_k \frac{\partial C_k}{\partial x} + \frac{Z_k C_k}{\sqrt{s^2+1}} \frac{\partial s}{\partial x} \\
D_k &= D_k^* + D_k^{+/-} \frac{N_k}{N_i} + D_k^{++/--} \frac{N_k}{N_i}^2 \\
s &= \sum_{k=1}^{N} \frac{Z_k C_k}{2N_i} \\
C(0,t) &= C_0 \frac{\partial C}{\partial x}(x >> 0, t) = 0
\end{aligned}$$

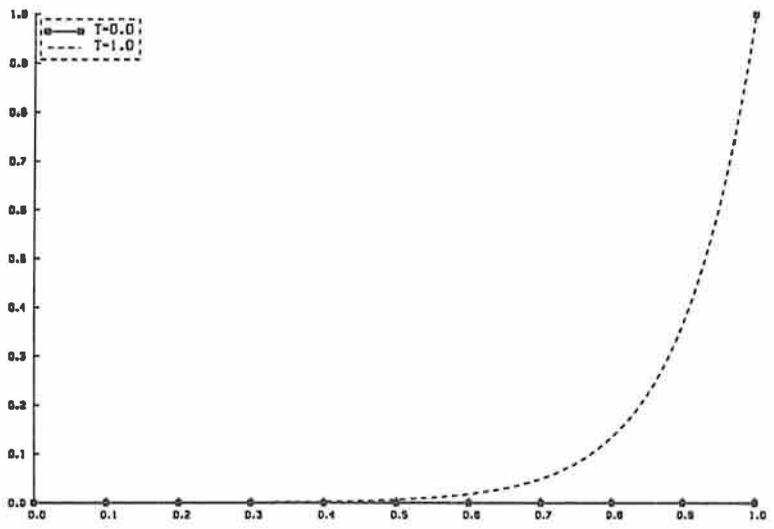
The initial condition for C is a Gaussian or Pearson distribution.
Source: A. van Run, Philips Research Laboratories, Eindhoven.

2.3 The need for adaptive meshing

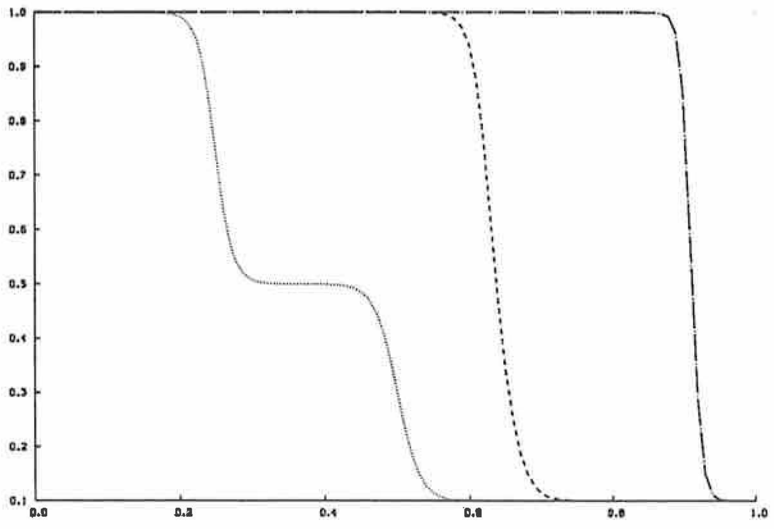
In the previous section the diverse nature of parabolic problems was demonstrated. It was seen that frequently in such problems the solution included regions where the physical variable was rapidly varying. Such situations arose from the presence of boundary layers and/or propagating shocks. To bring to mind some of these phenomena Figure 2.2 (a) shows the solution to Problem 5 for $\frac{k}{\epsilon} = 10$ and T=0,1. Figure 2.2 (b) shows the solutions of Burgers' equation (problem 7) for the overtaking shocks case at t=0,0.5 and 1.0 for $\epsilon = 0.003$.

Figures 2.2 (a) and (b) indicate that for parabolic problems regions of rapid variation in the solution may be fixed in time or transient and may not be confined to one particular location. This observation of course makes no difference if the problems can be solved analytically. However, if we now consider the numerical solution of parabolic problems using currently available methods such features as shocks and boundary layers become problematic.

In the following chapter the present-day methods for the solution of problems of the form (2.5) will be outlined. A numerical method for such a problem involves replacing the continuous equation with a discrete set of approximations (discretisation) and using these approximations deriving a solution



(a)



(b)

Figure 2.2: (a) Solution of Problem 5 for $k/\epsilon = 10$ at t=0,1. (b) Solution of Burgers' equation (Problem 7) for ϵ=0.003 at t=0, 0.5, 1.0.

which is consistent with the original equation. The numerical method should be sufficiently accurate in order to resolve the main features of the solution to the continuous problem.

The process of discretising equation (2.5) can be broken up into both spatial and temporal discretisations. In both situations the continuous derivatives should be accurately resolved by the discrete approximation. Chapter 3 deals with these approximations in detail. Associated with each approximation is an *order of accuracy*. For instance Finite Differences might be used to discretise equation (2.5) in space and time by imposing a computational mesh on the region $a \leq x \leq b, \quad t \geq 0$. As the overall number of mesh points are increased and the corresponding mesh spacings are decreased the numerical solution is required to approach the actual solution of the continuous problem.

Finite Difference approximations result from the linear combination of truncated Taylor series and the associated discretisation errors are easily determined. Similarly expressions for the discretisation error of Finite Element methods can also be obtained. The truncated terms involve higher derivatives of the dependent variable. Generally the local discretisation error is proportional to these higher derivatives and some power of the mesh spacing. Thus the accuracy of such approximations depends not only upon the mesh size but also on the form of the solution.

For instance if the problems of Figures 2.2 (a) and (b) were discretised uniformly over their entire domain then the error of the numerical method would be greater in regions where the derivatives of the solution are large corresponding to regions of rapid variation. Thus for such problems involving steep gradients the resolution of the solution must be maintained by imposing a finer grid. If the grid is kept uniform and the mesh spacing reduced in order to resolve a steep front in the solution then the number of grid points will increase enormously. Outside the region of rapid variation such a high concentration of grid points may be unnecessary since the derivatives of the solution are small in such regions. Thus employing a ***uniform but fine*** grid in order to minimise the maximum truncation error would appear to be computationally expensive in such cases. In the next chapter some numerical experiments are performed using Burgers' equation that bear this out.

It would obviously be far superior to impose a non-uniform computational mesh in space and time in which the mesh could be concentrated in areas of rapid variation. This is the basic principle behind the variable-mesh approach which will be discussed further in Chapter 4. Such a mesh would tend to equidistribute the truncation error throughout the problem domain. In the time discretisation of parabolic partial differential equations this principle has been used with much success for several years and the area of time-stepping is well developed. We will therefore concentrate on the question of spatial gridding. Attempting to equidistribute the truncation error over the spatial domain using a fixed non-uniform grid might lead to problems since Figure 2.2 reminds us that regions of rapid variation in the solution might shift as the problem evolves. Thus to resolve the evolving solution would require a ***variable mesh strategy***. This would involve the periodic adaptation of the mesh as the problem evolves in the following way.

- Determination of regions in the spatial grid where the solution is in a state of rapid variation.
- Assigning a suitable non-uniform grid in an attempt to equidistribute the truncation error.

Having observed the diversity of parabolic problems and identified the need for adaptive meshing we now have a notion of how such techniques might be implemented. Chapter 4 reviews some of the currently popular adaptive space-meshing strategies for parabolic problems.

Chapter 3

Standard Numerical Solution Procedures

In this chapter standard numerical methods for the solution of equation (2.5) are examined. These methods are similar in that firstly the continuous problem is discretised in space (semidiscretisation) and then the resulting system of ODEs is integrated in time.

Two main types of semidiscretisation methods prevail. These are the Finite Difference and Finite Element methods which are outlined in section (3.1). When Finite Differences are used the method is termed the *method of lines (MOL)*. The suitability of these two classes of semidiscretisation methods for the solution of the boundary layer and/or propagating shock problems of Chapter 2 are investigated.

The problem of solving the system of ODEs resulting from the semidiscretisation of parabolic equations is treated in section 3.2. The nature of the ODE system in such cases warrants the use of ODE integration methods which are suitable for so called "stiff" problems. A review of stiff ODE integration methods is presented.

Presently available PDE software embodies automatic semidiscretisation and ODE integration as described above. Two popular packages for PDE problems are reviewed in section 3.3. Such packages presume negligible spatial errors and the user must ensure that the spatial approximation and grid are sufficient to accurately represent the solution of the problem.

Section 3.4 examines the results of applying one such package to the solution of Burgers' equation (Problem 7, Chapter 2) on a uniform spatial grid. For solutions involving steep gradients, and especially those which are in motion, the results are inaccurate. To obtain reasonable accuracy the uniform grid must be smaller than is computationally practical. It is obvious that the difficulties encountered in solving such problems as Burgers' equation stem primarily from the ineffectiveness of the spatial grid and to a lesser extent on the spatial approximation. Even the use of non-uniform spatial grids which are fixed in time do not appreciably improve the results. The requirement for a spatial grid which adapts in time is clearly demonstrated.

3.1 Methods of Semidiscretisation

The methods of semidiscretising equations of type (2.5) are now examined. For spatial discretisation the most general approach is to divide the spatial interval [a,b] into N contiguous mesh spacings thus forming a general non-uniform grid

$$\Pi_N : a = x_0 < x_1 < x_2 < \ldots < x_N = b$$

Such a grid is shown in Figure 3.1

If the spatial terms of (2.5) are discretised using the grid Π_N then each PDE reduces to an ODE at each mesh point which evolves in the time direction. Each equation in (2.5) therefore reduces to a system of N+1 ODEs. This is why the method of semidiscretisation is often referred to as the method of lines (MOL). The two main methods of semidiscretisation are the Finite Difference and Finite Element methods, which shall now be examined.

3.1.1 Finite Differences

Finite Difference approximations arise when continuous derivatives are approximated by truncated Taylor series. This involves the imposition of a grid and the replacement of the continuous derivative at each grid point by

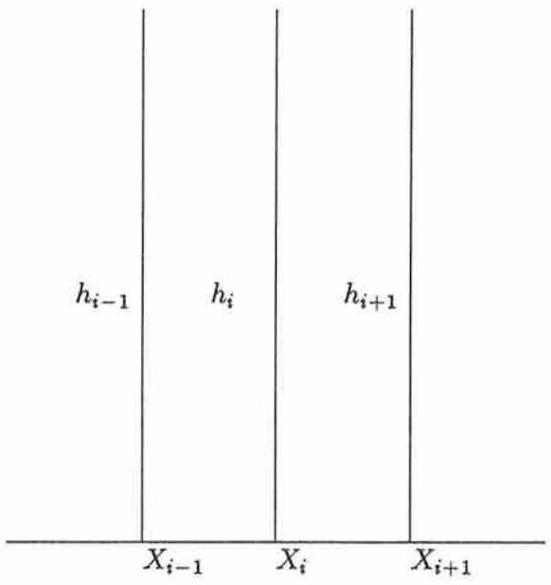


Figure 3.1: General non-uniform grid on the interval [a,b].

a linear combination of functional values at adjacent points.

For example, in the Finite Difference approximation of the spatial derivatives in (2.5), assume that the spatial interval [a,b] has been discretised using the general non-uniform mesh Π_N above. The following replacement of y_i' is possible in terms of y values at adjacent mesh points.

$$y_i' = \alpha y_{i+1} + \beta y_i + \gamma y_{i-1}$$

Using the notation

$$p = \Delta x_{i+1} = x_{i+1} - x_i \qquad\qquad q = \Delta x_i = x_i - x_{i-1}$$

the values $y_{i\pm1}$ may be expressed as Taylor series about y_i as follows.

$$y_{i-1} = y_i - qy_i' + \frac{q^2}{2!}y_i'' - \frac{q^3}{3!}y_i^{(iii)} + \frac{q^4}{4!}y_i^{(iv)} - \frac{q^5}{5!}y_i^{(v)} + \ldots \quad (3.1)$$

$$y_{i+1} = y_i + py_i' + \frac{p^2}{2!}y_i'' + \frac{p^3}{3!}y_i^{(iii)} + \frac{p^4}{4!}y_i^{(iv)} + \frac{p^5}{5!}y_i^{(v)} + \ldots \quad (3.2)$$

or, more generally,

$$y_{i-1} = \sum_{k=0}^{\infty} \frac{(-1)^k q^k}{k!} \frac{\partial^k y_i}{\partial x^k} \quad and \quad y_{i+1} = \sum_{k=0}^{\infty} \frac{p^k}{k!} \frac{\partial^k y_i}{\partial x^k}$$

Solving (3.1) and (3.2) for y_i' gives

$$y_i' = \frac{1}{q}\left\{y_i - y_{i-1} + \frac{q^2}{2!}y_i'' - \frac{q^3}{3!}y_i^{(iii)} + \frac{q^4}{4!}y_i^{(iv)} - \frac{q^5}{5!}y_i^{(v)} + \ldots\right\} \quad (3.3)$$

$$y_i' = \frac{1}{p}\left\{y_{i+1} - y_i + \frac{p^2}{2!}y_i'' - \frac{p^3}{3!}y_i^{(iii)} + \frac{p^4}{4!}y_i^{(iv)} - \frac{p^5}{5!}y_i^{(v)} + \ldots\right\} \quad (3.4)$$

Adding p times (3.3) and q times (3.4) causes the second derivative terms to vanish as follows

$$py_i' = \frac{p}{q}\{y_i - y_{i-1}\} + \frac{pq}{2!}y_i'' - \frac{pq^2}{3!}y_i^{(iii)} + \frac{pq^3}{4!}y_i^{(iv)} - \frac{pq^4}{5!}y_i^{(v)} + \ldots$$

$$qy_i' = \frac{q}{p}\{y_{i+1} - y_i\} - \frac{qp}{2!}y_i'' - \frac{qp^2}{3!}y_i^{(iii)} + \frac{qp^3}{4!}y_i^{(iv)} - \frac{qp^4}{5!}y_i^{(v)} + \ldots$$

$$\begin{aligned}(p+q)y_i' &= \frac{q}{p}\{y_{i+1}-y_i\}+\frac{p}{q}\{y_i-y_{i-1}\}-y_i^{(iii)}\frac{qp^2+pq^2}{3!}-y_i^{(iv)}\frac{qp^3-pq^3}{4!}\\ &\quad -y_i^{(v)}\frac{qp^4-pq^4}{5!}+\ldots\end{aligned}$$

$$\begin{aligned}y_i' &= \frac{1}{p+q}\left\{\frac{q}{p}\{y_{i+1}-y_i\}+\frac{p}{q}\{y_i-y_{i-1}\}\right\}-\frac{pq}{6}y_i^{(iii)}-\frac{pq}{24}(p-q)y_i^{(iv)}\\ &\quad -\frac{pq}{120}(p^2-pq+q^2)y_i^{(v)}+\ldots \qquad (3.5)\end{aligned}$$

Thus y_i' becomes

$$y_i' = \frac{1}{p+q}\left\{\frac{q}{p}\{y_{i+1}-y_i\}+\frac{p}{q}\{y_i-y_{i-1}\}\right\}+E_1 \qquad (3.6)$$

where

$$E_1 = -\frac{pq}{6}y_i^{(iii)}-\frac{pq}{24}(p-q)y_i^{(iv)}-\frac{pq}{120}(p^2-pq+q^2)y_i^{(v)}+\ldots$$

Putting this in the standard form (3.1.1) gives

$$y_i' = \{\frac{q}{p(p+q)}\}y_{i+1}+\{\frac{p^2-q^2}{pq}\}y_i+\{-\frac{p}{q(p+q)}\}y_{i-1}+E_1$$

In the case of a uniform grid (p=q=h) this expression reduces to

$$y_i' = \{\frac{1}{2h}\}y_{i+1}+\{-\frac{1}{2h}\}y_{i-1} \qquad (3.7)$$

where E_1 becomes

$$E_1 = -\frac{h^2}{6}y_i^{(iii)} - \frac{h^4}{120}y_i^{(iv)} + \ldots$$

This is the familiar second order accurate, centered difference approximation of y_i'. In a similar fashion an approximation of y_i'' in the form (3.1.1) is possible.

$$y_i'' = \alpha y_{i+1} + \beta y_i + \gamma y_{i-1}$$

Again, solving equations (3.1) and (3.2), this time for y_i'', gives

$$y_i'' = \frac{2}{q^2}\left\{y_{i-1} - y_i + qy_i' + \frac{q^3}{3!}y_i^{(iii)} - \frac{q^4}{4!}y_i^{(iv)} + \frac{q^5}{5!}y_i^{(v)} + \ldots\right\} \quad (3.8)$$

$$y_i'' = \frac{2}{p^2}\left\{y_{i+1} - y_i - py_i' - \frac{p^3}{3!}y_i^{(iii)} - \frac{p^4}{4!}y_i^{(iv)} - \frac{p^5}{5!}y_i^{(v)} + \ldots\right\} \quad (3.9)$$

Adding q times (3.8) and p times (3.9) and solving for y_i'' gives

$$\begin{aligned}(p+q)y_i'' &= \frac{2}{q}\{y_{i-1} - y_i\} + 2\left\{y_i' + \frac{q^2 y_i^{(iii)}}{3!} - \frac{q^3 y_i^{(iv)}}{4!} + \frac{q^4 y_i^{(v)}}{5!} + \ldots\right\} \\ &+ \frac{2}{p}\{y_{i+1} - y_i\} + 2\left\{-y_i' - \frac{p^2 y_i^{(iii)}}{3!} - \frac{p^3 y_i^{(iv)}}{4!} - \frac{p^4 y_i^{(v)}}{5!} + \ldots\right\}\end{aligned}$$

$$\begin{aligned}(p+q)y_i'' &= \frac{2}{q}\{y_{i-1} - y_i\} + \frac{2}{p}\{y_{i+1} - y_i\} + \frac{y_i^{(iii)}}{6}(q^2 - p^2) + \frac{y_i^{(iv)}}{12}(-q^3 - p^3) \\ &+ \frac{y_i^{(v)}}{60}(q^4 - p^4) + \ldots\end{aligned}$$

Thus y_i'' becomes

$$y_i'' = \frac{2}{p+q}\left\{\frac{1}{p}\{y_{i+1} - y_i\} - \frac{1}{q}\{y_i - y_{i-1}\}\right\} + E_2 \qquad (3.10)$$

where

$$E_2 = -\frac{p-q}{6}y_i^{(iii)} - \frac{p^2 - pq + q^2}{12}y_i^{(iv)} - \frac{p^3 - p^2q + q^2p + q^3}{60}y_i^{(v)} + \dots$$

Putting this in the standard form (3.1.1) gives

$$y_i'' = \{\frac{2}{p(p+q)}\}y_{i+1} + \{-\frac{2}{pq}\}y_i + \{-\frac{2}{q(p+q)}\}y_{i-1} + E_2$$

Again, in the case of a uniform grid (p=q=h) this expression reduces to

$$y_i'' = \{\frac{1}{h^2}\}y_{i+1} + \{-\frac{2}{h^2}\}y_i + \{\frac{1}{h^2}\}y_{i-1} + E2 \qquad (3.11)$$

where E_2 becomes

$$E_2 = -\frac{h^2}{12}y_i^{(iv)} + \dots$$

Tables 3.1 and 3.2 summarise the above Finite Difference approximations for both uniform and non-uniform grids.

$y'_i = \alpha y_{i+1} + \beta y_i + \gamma y_{i-1}$		
	uniform	non-uniform
α	$\frac{1}{2h}$	$\frac{q}{p(p+q)}$
β	0	$\frac{p-q}{pq}$
γ	$-\frac{1}{2h}$	$-\frac{p}{q(p+q)}$
T.E. (leading term)	$-\frac{h^2}{6}y^{(iii)}$	$-\frac{pq}{6}$

Table 3.1: Discretisation coefficients and leading term of the truncation error for approximation of $\frac{\partial U_i}{\partial x}$ on both uniform and non-uniform grids.

The Finite Difference formulae treated up to now have been *centered* approximations. In practice, however, such centered approximations are not always useful.

Consider once again the steady state of Problem 5 of Chapter 2.

$$\epsilon \frac{\partial^2 U}{\partial x_2} - k\frac{\partial U}{\partial x} = 0 \tag{3.12}$$

subject to the boundary conditions U(0)=0 and U(1)=1. The solution of this equation was seen to be

$$U = \frac{e^{kx/\epsilon} - 1}{e^{k/\epsilon} - 1} \tag{3.13}$$

$y''_i = \alpha y_{i+1} + \beta y_i + \gamma y_{i-1}$		
	uniform	non-uniform
α	$\frac{1}{h^2}$	$\frac{2}{p(p+q)}$
β	$-\frac{2}{h^2}$	$-\frac{2}{pq}$
γ	$\frac{1}{h^2}$	$\frac{2}{q(p+q)}$
T.E. (leading term)	$-\frac{h^2}{12}y_i^{(iv)}$	$-\frac{p-q}{6}y_i^{(iii)}$

Table 3.2: Discretisation coefficients and leading term of the truncation error for approximation of $\frac{\partial^2 U_i}{\partial x^2}$ on both uniform and non-uniform grids.

Let us now examine the effectiveness of the centered approximations applied to this equation. For simplicity a uniform grid $x_i = (i-1)h$, i=1, ...,N+1 is presumed. Applying the approximations (3.7) and (3.11) results in the following algebraic equation at internal nodes.

$$\frac{\epsilon[U_{i+1} - 2U_i + U_{i-1}]}{h^2} - \frac{k(U_{i+1} - U_{i-1})}{2h} = 0 \qquad i = 2, \dots, N$$

or

$$U_{i+1} - 2U_i + U_{i-1} - \frac{kh}{2\epsilon}[U_{i+1} - U_{i-1}] = 0 \qquad i = 2, \dots, N \qquad (3.14)$$

with U(1)=0 and U(N+1)=1 at the boundary points.

This equation is a homogeneous linear difference equation of order two. This can be solved as follows.

Letting $U_i = r^i$ the equation becomes

$$\begin{aligned} r^{i+1} - 2r^i + r^{i-1} - \frac{kh}{2\epsilon}\left(r^{i+1} - r^{i-1}\right) &= 0 & (3.15) \\ \left(r^2 - 2r + 1\right) - \frac{kh}{2\epsilon}\left(r^2 - 1\right) &= 0 \end{aligned}$$

Denoting $\frac{kh}{2\epsilon}$ by β implies

$$\begin{aligned} \left(r^2 - 2r + 1\right) - \beta\left(r^2 - 1\right) &= 0 \\ r^2(1-\beta) + r(-2) + (1+\beta) &= 0 \end{aligned}$$

The solutions to this quadratic are

$$\begin{aligned} r &= \frac{2 \pm \sqrt{4 - 4(1-\beta^2)}}{2(1-\beta)} \\ &= \frac{1}{1-\beta} \pm \frac{\beta}{1-\beta} \end{aligned}$$

The principle of *superposition* allows a general solution to (3.15) to be constructed as follows.

$$\begin{aligned} U_i &= A_0\left(\frac{1}{1-\beta} + \frac{\beta}{1-\beta}\right)^i + B_0\left(\frac{1}{1-\beta} - \frac{\beta}{1-\beta}\right)^i \\ &= A_0\left(\frac{1+\beta}{1-\beta}\right)^i + B_0 & (3.16) \end{aligned}$$

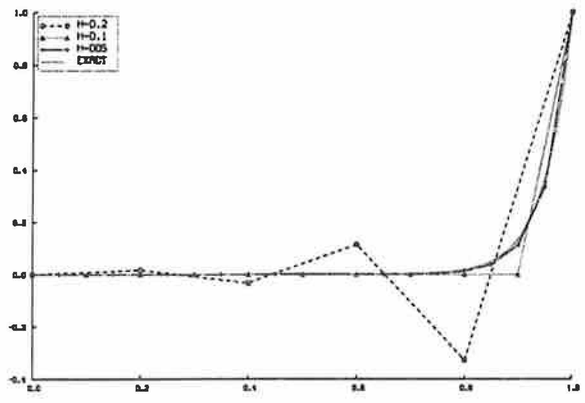


Figure 3.2: Comparison of exact and centered difference solutions for $k/\epsilon =$ 20 and h=0.05, 0.1 and 0.2.

The exact solution of the continuous problem (3.13) and the exact solution of the difference equation (3.16) can be compared. Figure 3.2 compares the two solutions for $k/\epsilon = 20$ and h=0.05, 0.1 and 0.2.

For h=0.05 (corresponding to $\beta = 0.5$) the solution of the difference equation agrees reasonably well with the exact solution of continuous problem. The solution corresponding to $\beta = 1$ is accurate except in the boundary layer and for $\beta = 2$ the solution is both oscillatory and inaccurate.

By examining (3.16) it can be seen that in order to guarantee a non-oscillatory solution the term in brackets must be positive. Thus for a non-oscillatory solution to the difference equation we require

$$\beta \leq 1 \tag{3.17}$$

This restriction is often referred to as a *cell Reynolds number* limitation. In order to avoid oscillations the grid spacing must be restricted in size so that (3.17) is satisfied. This is equivalent to requiring that

$$\frac{kh}{\epsilon} \leq 2$$

A popular technique for avoiding oscillatory solutions is the method of ***up-winding*** [19]. This involves the approximation of the convective term $\frac{\partial U}{\partial x}$ by a ***one sided*** Finite Difference approximation which is first order accurate[1]. For example in the above problem where the boundary layer is at x=1 the ***backward difference*** approximation of $\frac{\partial U}{\partial x}$

$$(\frac{U_i - U_{i-1}}{h})$$

may be used as follows.

$$U_{i+1} - 2U_i + U_{i-1} - \frac{kh}{\epsilon}\left[U_i - U_{i-1}\right] = 0 \tag{3.18}$$

This difference equation is solved in the same way as equation (3.14) to give

$$U_i = B_0\,(1+\beta)^i + A_0, \qquad \beta = \frac{kh}{\epsilon} \tag{3.19}$$

Figure 3.3 compares the exact solution of the upwinded difference scheme (3.19) and the previously derived scheme (3.14) with the exact solution of the continuous problem (3.13) for h=0.2 and $k/\epsilon = 20$. The upwind scheme exhibits non-oscillatory behaviour but is also inaccurate. It is instructive to write the convective term in the upwinded difference scheme (3.18) as follows.

[1]The reason why this technique is called *upwinding* is because only information upwind of node k is transmitted to node k by convection. This technique was first used in weather prediction models; hence the name.

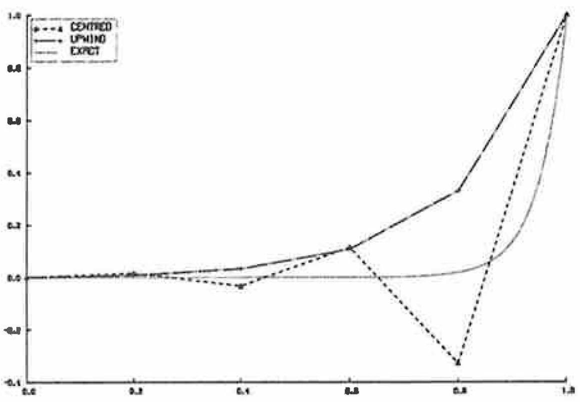


Figure 3.3: Comparison of upwind and centered difference solutions with the exact solution for $k/\epsilon = 20$ and $h = 0.2$

$$
\begin{aligned}
k\frac{\partial U}{\partial x} &= \frac{k}{h}\left[U_{i+1} - U_i\right] \\
&= \frac{k}{h}\left[U_{i+1} - U_{i-1}\right] - \frac{k}{2h}\left[U_{i+1} - 2U_i + U_{i-1}\right]
\end{aligned}
$$

This corresponds to the second order centered approximation of the convective term plus an additional diffusive term. Thus the upwinded scheme is equivalent to solving equation (3.12) with an effective diffusivity of

$$
\epsilon\left[1+\beta\right], \qquad (\beta = \frac{kh}{2\epsilon})
$$

This extra diffusive term, encurred by the use of upwinding, is known as ***numerical diffusion***. In the next section an approximation of the convection-diffusion equation shall be derived, using Finite Elements, which allows upwinding but also some control over the resulting numerical diffusion.

The semidiscretisations discussed above when applied to the solution of the linear problems of Chapter 2 result in systems of coupled ODEs. If the number of original PDEs is $\mathcal{N}$ and the number of spatial grid intervals is N then the dimension of the resulting ODE system will be $(N+1)*\mathcal{N}$. If the methods are applied to the nonlinear problems of Chapter 2 the result is still a system of $(N+1)*\mathcal{N}$ ODEs. Thus as far as semidiscretisation is concerned the problem in question may just as easily be nonlinear as linear.

Before proceeding further with the methods of Finite Differences, consideration is now given to the approximation of boundary conditions for problems of type (2.5). Consider the general boundary conditions

$$\begin{aligned} U_i &= f_i(x) \quad t=0 \\ p_i(t)U_i + q_i(t)\frac{\partial U_i}{\partial x} &= r(U_i,t) \quad x=a,b \quad t>0 \\ & i = 1,2\ldots\mathcal{N}. \end{aligned}$$

Two main cases arise

Dirichlet boundary conditions: Here q(t)=0 at x=a,b and the solution value U_1 is obtained directly using

$$U_1 = \frac{r(t)}{p(t)}$$

Note: if q(t)=0 then r must be independent of U.

Since U_1 is a known function of time it is not necessary tc solve an ODE for the value of U_1 as the problem evolves. Thus in the case of Dirichlet boundary data the system of ODEs is reduced by $2*\mathcal{N}$. For a single Parabolic equation ($\mathcal{N}$=1) with Dirichlet data if the second order centered Finite Difference semidiscretisation is chosen then the original PDE reduces to

$$
\begin{aligned}
U_1 &= \frac{r_1(t)}{p_1(t)} \\
\frac{\partial U}{\partial t} &= f(U_{i-1}, U_i, U_{i+1}) \qquad , i = 2, \ldots, N \\
U_{N+1} &= \frac{r_2(t)}{p_2(t)}
\end{aligned}
$$

General boundary conditions: Here $q(t) \neq 0$ at x=a,b and the solution value U_1 is not explicitly available. The spatial operator can be evaluated at the boundaries leading to ODEs for the boundary values. The first order derivative at the boundary is available from the boundary condition as

$$
\left(\frac{\partial U}{\partial x}\right)_{x=a,b} = \frac{r(U,t) - p(t)U}{q(t)}
$$

However the second derivative at the boundary cannot be approximated using central differences since values outside the boundaries are not available for the approximation. Forward differences at x=a and backward differences at x=b must be used in such cases. The representation of the spatial operator at the boundaries is therefore only first order accurate. For a single Parabolic equation ($\mathcal{N}$=1) with general boundary data the second order centered Finite Difference semidiscretisation leads to the following system of ODEs.

$$
\frac{\partial U}{\partial t} = f(U_{i-1}, U_i, U_{i+1}) \qquad i = 1, \ldots, N+1
$$

In the case of general boundary conditions, the order of accuracy of the finite difference scheme suffers somewhat at the boundaries. This is one of the main difficulties encountered in the solution of parabolic PDEs. In Chapter 4 the improvement of boundary representation by using non-uniform grids will be investigated. The main reason for the popularity of the centered and upwinded Finite Difference semidiscretisations is because the Jacobian of the resulting ODE system is always ***tridiagonal*** for a single PDE and ***block tridiagonal*** for a system of PDEs. This means that in the solution of the ODE system using implicit methods, only tridiagonal linear systems of algebraic equations occur. Efficient algorithms are available for the solution of such systems. See [47], [52] and [30].

For higher order Finite Difference approximations the problems of representing boundary conditions are even greater. For example a centered ***fourth*** order approximation for $\frac{\partial^2 U}{\partial x^2}$ based on a five point formula is

$$\frac{\partial^2 U}{\partial x^2} = \frac{-U_{i-2} + 16U_{i-1} - 30U_i + 16U_{i+1} - U_{i+2}}{12h^2} \tag{3.20}$$

This fourth order approximation is of little practical since fourth order accuracy will not be possible at nodes adjacent to the boundaries. This is because the computational molecule for this semidiscretisation spans five nodal values.

Due to these consequences, higher order Finite Difference schemes have been sought which maintain the tridiagonal nature of the spatial operator. Ciment et. al. [9] derived a fourth order semidiscretisation of this kind for the convection-diffusion equation. Rather than approximate the spatial derivatives independently to fourth order the ***operator compact implicit*** method establishes a fourth order accurate relationship between the spatial operator

$$L(U) = a(x)\frac{\partial^2 U}{\partial x_2} + b(x)\frac{\partial U}{\partial x} \tag{3.21}$$

and the function U on three adjacent mesh points. This is similar to the approach used in the Galerkin Finite Element method which will be discussed in 3.1.2. This relationship is as follows

$$q_i^+ \left(L(U)\right)_{i+1} + q_i^0 \left(L(U)\right)_i + q_i^- \left(L(U)\right)_{i-1} = \frac{r_i^+ U_{i+1} + r_i^0 U_i + r_i^- U_{i-1}}{h^2} \tag{3.22}$$

where

$$\begin{aligned}
q_i^+ &= 6a_i a_{i-1} + h(5a_{i-1}b_i - 2a_i b_{i-1}) - h^2 b_i b_{i-1} \\
q_i^- &= 6a_i a_{i+1} - h(5a_{i+1}b_i - 2a_i b_{i+1}) - h^2 b_i b_{i+1} \\
q_i^0 &= 4[15a_{i+1}a_{i-1} - 4h(a_{i+1}b_{i-1} - b_{i+1}a_{i-1}) - h^2 b_{i+1}b_{i-1}] \\
r_i^+ &= \frac{1}{2}[q_i^+(2a_{i+1} + 3hb_{i+1}) + q_i^0(2a_i + hb_i) + q_i^-(2a_{i-1} - hb_{i-1})] \\
r_i^- &= \frac{1}{2}[q_i^+(2a_{i+1} + hb_{i+1}) + q_i^0(2a_i - hb_i) + q_i^-(2a_{i-1} - 3hb_{i-1})] \\
r_i^0 &= -(r_i^+ + r_i^-)
\end{aligned}$$

This is written as

$$\frac{Q^{-1}R}{h^2}U(x_i) = L(U_i) + O(h^4)$$

where Q and R are tridiagonal displacement operators. A Taylor series analysis of (3.22) allows the coefficients r_i and q_i to be determined.

For Dirichlet data fourth order accuracy is maintained across the spatial interval $a \leq x \leq b$ with a computational molecule spanning only three adjacent nodes. This is identical to the second order Finite Difference replacement of (3.21). In the case of general boundary conditions, fourth order accurate expressions involving the boundary values can be obtained using the boundary nodes and the two adjacent nodes. This alters the tridiagonal nature of the ODE system resulting from the semidiscretisation but can easily be handled by preprocessing the linear system prior to using a conventional tridiagonal solver.

3.1.2 Finite Elements

Finite Element methods result from an integral representation of the evolutionary equation. These methods involve building an approximation of the exact solution using linear combinations of basis functions in each subregion (finite element) of the computational grid.

One of the most popular Finite Element methods for the discretisation of elliptic problems and the spatial discretisation of parabolic problems is the Galerkin Finite Element method [18]. This method is a special case of the more general class of weighted residual methods.

Consider the general parabolic partial differential equation

$$\frac{\partial U}{\partial t} = A(U) \qquad (x \epsilon \mathcal{R}, t > 0) \tag{3.23}$$

The function U(x,t) which satisfies (3.23) is termed a classical solution. Problems expressed in this form may be solved using classical Finite Element methods if the function U is also the solution to some variational problem. See [38], Chapter 2. It is however possible to apply Finite Element methods to problems of the form (3.23), for which variational problems do not exist, by considering the **weak** form of the problem

$$(U_t, w) \quad = \quad (A(U), w) \tag{3.24}$$

where w is some function and where the inner product (a, b) is defined as

$$(a, b) = \int_{\mathcal{R}} a(x)b(x)dx$$

Solutions to this problem are generally less continuous than their classical counterparts and may not even be distinct. However the use of the weak formulation greatly extends the use of Finite Element methods.

The method of weighted residuals generates an approximate solution of (3.23) of the form

$$U(x,t) = U_0(x,t) + \sum_{j=1}^{N} a_j(t)\phi_j(x) \tag{3.25}$$

where $\phi_j(x)$ form a linearly independent set of known analytic functions. These are often called ***trial functions*** and equation (3.25) the ***trial solution***. The functional U_0 is chosen to satisfy the boundary conditions. Equation (3.25) suggests that equation (3.24) reduces to a set of ordinary differential equations in t. The coefficients a_j in (3.25) need to be determined and one method is to set the inner product of the weighted residual, $R = \frac{\partial U}{\partial t} - A(U)$, to zero

$$(R, w_j(x)) = 0 \qquad j = 1, \ldots, N \tag{3.26}$$

This is where the method takes its name and w_j above is referred to as the *weight* or *test* function.

In the two sections to follow, the Galerkin and Collocation Finite Element methods are examined. Both of these methods belong to the class of weighted residual methods.

Galerkin Finite Element methods.

In this method the weight function is chosen from the same family as the trial functions. The inner product of the weighted residual thus becomes

$$(R, \phi_j(x)) = 0 \quad j = 1, \ldots, N \quad a \leq x \leq b \tag{3.27}$$

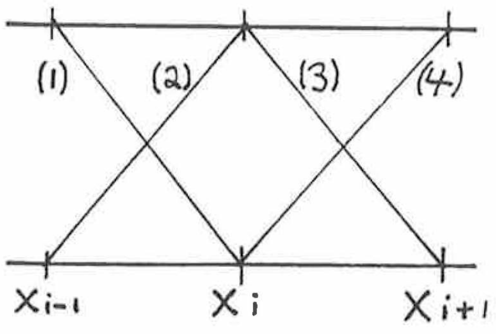
Solving this system allows the constants a_j in equation (3.25) to be determined. It is essential that the functions $\phi_i(x)$ be linearly independent since otherwise the system of equations represented by (3.27) will become ill-conditioned. This occurs for large N in the traditional Galerkin method where the trial functions are defined over the entire spatial domain, as demonstrated by Fletcher [18]. The use of piecewise trial spaces guarantees that each trial function will only have local support and thus the system (3.27) will remain well conditioned, even for large N. The use of piecewise polynomial trial spaces with the Galerkin method constitutes the *Galerkin Finite Element* method.

For example, consider the piecewise linear trial functions shown in Figure 3.4. Let us apply the Galerkin Finite Element method with the linear trial functions of Figure 3.4 to the solution of Problem 1, Chapter 2.

$$\begin{aligned}
\frac{\partial U}{\partial t} &= \frac{1}{\pi^2}\frac{\partial^2 U}{\partial x^2} \qquad 0 \leq x \leq 1 \\
U(0,t) &= U(1,t) = 0 \\
U(x,0) &= \sin(\pi x) \\
U(x,t) &= e^{-t}\sin(\pi x)
\end{aligned}$$

Introducing the approximate solution

$$U(x,t) = \sum_{j=1}^{N} a_j(t)\phi_j(x)$$



$$
\begin{aligned}
(1) \rightarrow \phi &= \frac{x_i - x}{x_i - x_{i-1}} \\
(2) \rightarrow \phi &= \frac{x - x_{i-1}}{x_i - x_{i-1}} \\
(3) \rightarrow \phi &= \frac{x_{i+1} - x}{x_{i+1} - x_i} \\
(4) \rightarrow \phi &= \frac{x - x_i}{x_{i+1} - x_i}
\end{aligned}
$$

Figure 3.4: Piecewise linear (hat) trial functions.

The weak form of the problem is

$$\left(\frac{\partial U}{\partial t}, \phi_j\right) - \left(\frac{\partial^2 U}{\partial x^2}, \phi_j\right) \qquad j = 1, \ldots, N$$

Substituting the trial solution into the weak form of the problem followed by integration by parts gives

$$
\begin{aligned}
\sum_{i=1}^{N} \frac{dU_i}{dt} (\phi_i, \phi_j) - \frac{1}{\pi^2} \sum_{i=1}^{N} \frac{d^2 U}{dx^2} \phi_j dx &= 0 \qquad j = 1, \ldots, N \\
\sum_{i=1}^{N} (\phi_i, \phi_j) \frac{dU_i}{dt} - \frac{1}{\pi^2} \sum_{i=1}^{N} \left(\frac{d\phi_i}{dx}, \frac{d\phi_j}{dx}\right) U_i &= 0 \qquad j = 1, \ldots, N
\end{aligned}
$$

Now we introduce the linear trial functions ϕ_i which, for convenience, we will presume are defined on a uniform mesh of size h. This allows the inner products to be calculated.

For example (ϕ_i, ϕ_j) can be determined as follows. Referring to Figure 3.4

$$\begin{aligned}
(\phi_i, \phi_j) &= \int_{x_{i-1}}^{x_i} \left(\frac{x - x_{i-1}}{x_i - x_{i-1}}\right)^2 dx + \int_{x_i}^{x_{i+1}} \left(\frac{x_{i+1} - x}{x_{i+1} - x_i}\right)^2 dx \\
&= \frac{1}{h^2} \int_{x_{i-1}}^{x_i} \left(x^2 - 2xx_{i-1} + x_{i-1}^2\right) dx + \frac{1}{h^2} \int_{x_i}^{x_{i+1}} \left(x_{i+1}^2 - 2xx_{i+1} + x^2\right) dx \\
&= \frac{1}{3h^2} \left[(x_i - x_{i-1})^3 + (x_{i+1} - x_i)^3\right] \\
&= \frac{h^3}{3h^2} + \frac{h^3}{3h^2} \\
&= \frac{2h}{3}
\end{aligned}$$

Similarly the inner products (ϕ_i, ϕ_{i+1}) and (ϕ_i, ϕ_{i-1}) become

$$(\phi_i, \phi_{i+1}) = \frac{h}{6} = (\phi_i, \phi_{i-1})$$

and

$$\left(\frac{d\phi_i}{dx}, \frac{d\phi_i}{dx}\right) = \frac{2}{h}$$

Using these inner products the trial solution becomes

$$\frac{1}{6}\frac{dU_{i-1}}{dt}+\frac{2}{3}\frac{dU_i}{dt}+\frac{1}{6}\frac{dU_{i+1}}{dt}=\frac{1}{\pi^2}\frac{(U_{i-1}-2U_i+U_{i+1})}{h^2}$$

The right hand side of the above Finite Element discretisation is identical to the centered second order Finite Difference replacement of $\frac{\partial^2 U}{\partial x^2}$ previously derived in section 3.1.1. For this reason, Galerkin Finite Element methods are often regarded as an alternative method of deriving Finite Difference approximations. In the above formula, the time derivative is distributed over three adjacent nodes of the mesh. Thus the Galerkin method establishes a linear relationship between U and L(U)($=\frac{\partial U}{\partial t}$) on three adjacent grid points. The resemblance between the Galerkin method and the operator compact implicit Finite Difference method studied in section 3.1.1 is obvious. Swartz [49] examines some difference schemes which closely resemble Finite Element methods and Varah [52] demonstrates the equivalence of the well known *box scheme* of Keller [28] and the Galerkin Finite Element method. As pointed out by Hopkins [27], the main difference between the method of lines and Finite Element methods is that the former method reduces equation (3.23) to an ODE system of the form

$$\frac{\partial U}{\partial t}=f(U)$$

whereas the latter leads to

$$B\frac{\partial U}{\partial t}=f(U)$$

As in the case of Finite Differences the Galerkin Finite Element method can be used to semidiscretise convection-diffusion equations. Consider once again the steady state of Problem 5, Chapter 2.

$$\epsilon\frac{\partial^2 U}{\partial x^2}-k\frac{\partial U}{\partial x}=0 \qquad (3.28)$$

subject to the boundary conditions U(0)=0 and U(1)=1. Application of the Galerkin Finite Element method with piecewise linear trial functions on a uniform mesh results in the following difference equation

$$U_{i+1} - 2U_i + U_{i-1} - \frac{kh}{2\epsilon}\left(U_{i+1} - U_{i-1}\right) = 0$$

This is identical to the difference equation (3.14) produced by second order centered Finite Differences. Thus the two formula give equivalent approximation of the spatial terms in the convection-diffusion equation. A cell Reynolds number limitation therefore also exists for the Galerkin semidiscretisation. Upwinding is also a common method of improving the spatial resolution of Galerkin methods. Such a scheme is an example of the generalised or Petrov-Galerkin [18] formulation and, in contrast to the case of Finite Difference, oscillations at high Reynolds number can be avoided *without* a reduction in accuracy.

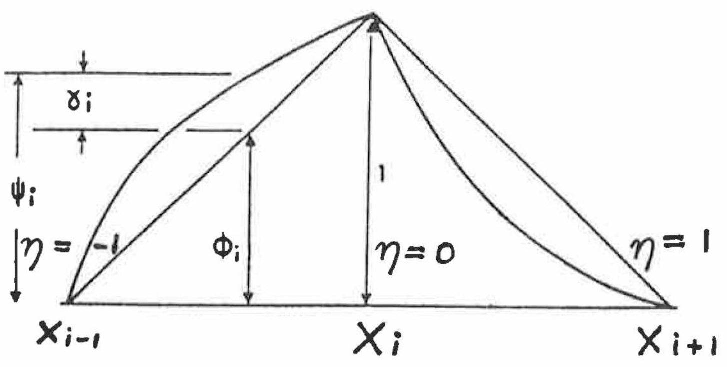
The upwind Galerkin Finite Element scheme is derived by introducing the following modified trial function

$$\psi_i(x) = \phi_i(x) + \alpha\gamma_i(x)$$

as shown in Figure 3.5.

The function γ is an antisymmetric quadratic perturbation function and α is a parameter which controls its influence. Using this trial function in the Galerkin formulation gives the following algebraic equation at internal nodes.

$$(1+\alpha)\left[U_{i-1} - 2U_i + U_{i+1}\right] - \frac{kh}{\epsilon}\left(U_i - U_{i-1}\right) = 0$$



$$\gamma_i = -3\eta(1-\eta) \text{ in element [i, i+1]}$$
$$\gamma_i = 3\eta(1+\eta) \text{ in element [i-1, i]}$$

where η is a local coordinate.

Figure 3.5: Trial function used in the ***upwind*** Galerkin Finite Element method.

When $\alpha = 0$ the scheme corresponds to the conventional Galerkin or Finite Difference semidiscretisation whereas $\alpha = 1$ gives full upwinding. Unlike the Finite Difference upwinded scheme, the use of the parameter α in the upwind Galerkin Finite Element scheme allows the numerical diffusion to be controlled.

Collocation methods.

In this weighted residual Finite Element method the weight function, $w_j(x)$ is chosen as

$$w_k(x) = \delta(x - x_j)$$

where δ is the Dirac *Delta* function.

Thus the residual $R_j = 0$ and (3.24) is satisfied exactly at a number of points $x_j \epsilon R (j = 1, \ldots, N)$ called collocation points. This set must include the boundary points. This property is also shared by most Finite Difference schemes.

The solution to $R(x_j) = 0 \quad (j = 1, \ldots, N)$ allows the coefficients a_j of the trial solution (3.25) to be determined. The main advantages of collocation are

- There are no inner products to integrate as in the Galerkin Finite Element method.
- The resultant semidiscretisation has fewer terms than in the Galerkin method.

The main disadvantages of the collocation method are

- The boundary and initial conditions must be consistent.
- It is necessary to use trial functions of at least the same order as the original differential equation.
- Collocation techniques are non-conservative and as such may be inappropriate for some problems based on conservation laws.

In section 3.3 the popular collocation software package PDECOL of Madsen and Sincovec [32] will be reviewed.

3.2 Temporal Integration

Having outlined the common methods of semidiscretisation for parabolic problems, consideration is now given to the solution of the resultant system of ordinary differential equations.

Section 3.2.1 examines the nature of the ODE system and introduces the notion of *stiffness*. The need to use stiff integration methods when dealing with equations of parabolic type is demonstrated.

The currently used methods for *stiff* systems of ODEs are reviewed. Software for ODEs is of a high quality and the integration of a problem is carried out in an automatic and optimal fashion. Some of the basic techniques used in the automatic integration of ODEs are outlined.

3.2.1 The nature of the ODE system and the notion of stiffness

In 3.1 various semidiscretisations from the class of Finite Difference and Finite Element methods were derived. Now let us consider the results of such semidiscretisations.

Consider a single parabolic equation of the form of equation (2.5) discretised in space on the uniform mesh

$$x_i = (i-1)h \qquad i = 1, \ldots, N+1 \qquad h = \frac{(b-a)}{N}$$

The result of such a semidiscretisation is a system of ODEs of the form

$$\frac{\partial U}{\partial t} = f(U,t) \tag{3.29}$$

in the case of Finite Differences, and

$$A\frac{\partial U}{\partial t} = f(U,t) \tag{3.30}$$

in the case of Finite Elements (with U(0) specified).

The semidiscretisation reduces an initial boundary value problem (IBVP) for a PDE to an initial value problem (IVP) for a system of ODEs. Based upon the grid given above these systems will be of dimension N+1. However, as mentioned in section 3.1.1, in the case of Dirichlet data, the ODEs corresponding to the boundary points reduce to the trivial cases

$$\frac{dU_1}{dt} = 0 \qquad \frac{dU_{N+1}}{dt} = 0$$

For generality however let us assume that the ODE system has dimension N+1. Section 3.2.2 outlines the presently used numerical methods for the solution of such systems. Firstly however it will be instructive to examine the properties of such systems resulting from the discretisation of parabolic problems. This will lead us to the notion of ***stiffness***.

As an example consider the semidiscretisation of the simple heat equation (Problem 1, Chapter 2)

$$\frac{\partial U}{\partial t} = \frac{1}{\pi^2}\frac{\partial^2 U}{\partial x^2} \qquad 0 \le x \le 1$$

Semidiscretising this equation using conventional second order centered Finite Differences produces the following ODE system

$$\frac{\partial U_i}{\partial t} = \frac{1}{\pi^2}\begin{bmatrix} -2 & 1 & & & & . & . & 0 & 0 \\ 1 & -2 & 1 & & & & & & 0 \\ & 1 & -2 & 1 & & & & & . \\ & & . & . & . & & & & . \\ & & & . & . & . & & & \\ & & & & . & . & . & & \\ . & & & & & . & . & . & \\ . & & & & & 1 & -2 & 1 & \\ 0 & & & & & & 1 & -2 & 1 \\ 0 & 0 & . & . & & & & 1 & -2 \end{bmatrix}\begin{bmatrix} U2 \\ . \\ . \\ . \\ . \\ . \\ . \\ . \\ . \\ UN \end{bmatrix} \tag{3.31}$$

The eigenvalues of the above matrix are given by Seward [45] and are real and negative.

$$\lambda_k = -\frac{2}{\pi^2 h^2}\left(1 + \cos\frac{k\pi}{N}\right) \qquad k = 1, \ldots, N$$

For large N the largest eigenvalue may be approximated by

$$\lambda_1 \approx -\frac{4N^2}{\pi^2}$$

and the smallest eigenvalue may be approximated by

$$\lambda_N \approx -1$$

The negative reciprocals of these eigenvalues correspond to the time constants of the ODE system. Thus for large N the system will possess disparate time constants. This may often lead to computational difficulties and in such cases the ODE system is termed *stiff*[2]. The conditions for stiffness to occur depend on the range of integration. Small decay time constants correspond to transient solution components. If the range of integration of the ODE system is restricted to the transient interval then the variation of the solution can be adequately represented using standard integration methods. If the range of integration is much larger than the transient interval, then the solution in this region is dominated by the slow components (corresponding to small eigenvalues). However small time steps must still be used in the integration in order to resolve the transient components in a stable manner, even though they hardly affect the solution.

Thus the stiffness of the system depends both upon the ratio of the maximum and minimum negative eigenvalues and the range of integration. An adequate measure of stiffness is the following index

$$S = \frac{t_{final}}{\tau_{min}} \quad or \quad S = t_{final}\lambda_{max}$$

where τ is the time step.

[2]The name "stiff" was introduced by Curtiss and Hirschfelder [12] because the servomechanism modelled by such an ODE system felt stiff.

A value of S $\approx$ 100 would be regarded as stiff and S $\approx$ 10 as non-stiff. For the above problem the stiffness index would be

$$\frac{4N^2}{\pi^2} t_{final}$$

Thus for N=100 and $t_{final} = \pi^2$ this problem would be stiff. For N=10 and $t_{final} = 1$ the problem would be mildly stiff.

From the numerical perspective stiffness arises when stability rather than accuracy dictates the time step. As pointed out by Seward et al. [45], and indicated by the above example, stiffness is associated with parabolic problems in which the diffusive process dominates. In order to guarantee an accurate solution to (3.29) or (3.30) we must anticipate the need for numerical methods suitable for stiff ODE systems. In the next section integration formulae suitable for stiff systems are examined.

3.2.2 Integration formulae for stiff ODE systems

Previously it was seen that stiffness depends on several factors including the type of problem and the numerical approach being used. It was also observed that stiffness may vary during the evolution of the solution. The measure of stiffness used above is primarily a qualitative one and precise measures are not in common use owing to the computational expense of calculating eigenvalues for a system of equations. A more modern approach to measuring stiffness is given in Bui et al. [5]. The presently used robust methods for ODE integration rely on formulae that cater for varying degrees of accuracy and stability requirements. By far the most popular of such methods are the linear multistep methods (LMM).

Linear multistep methods are formulae of the form

$$\sum_{k=0}^{K} \alpha_k u_{n+1-k} + h \sum_{k=0} \beta_k \dot{u}_{n+1-k} = 0$$

for solving the ODE system

$$\dot{u} = f(u,t) \tag{3.32}$$

where h is the step size and the coefficients α_k and β_k depend on h. By convention α_0 is chosen to be -1.

Two notable families of LMM exist, namely the ***Adams Moulton*** (AM) methods and the ***backward differentiation formulae*** (BDF) popularised by Gear [22]. The general form of the Adams Moulton method is

$$u_{n+1} = u_n + h \sum_{k=0}^{K-1} \beta_k \dot{u}_{n+1-k}$$

If β_k=0 in the above formula then u_{n+1} can be obtained explicitly. The first order explicit AM method is the well-known Euler method

$$u_{n+1} = u_n + h\dot{u}_{n+1} \qquad (\beta_0 = 0, \beta_1 = 1)$$

The second order AM formula is the *trapezoidal rule*

$$u_{n+1} = u_n + \frac{h}{2}\left(\dot{u}_{n+1} + \dot{u}_n\right) \qquad (\beta_0 = 0, \beta_1 = \frac{1}{2})$$

This method corresponds to the well known Crank Nicolson time discretisation for partial differential equations. See Mitchell and Griffiths [37].

The BD formulae are implicit multistep methods of the form

$$-\beta_0 h\dot{u}_{n+1} = -u_{n+1} + \sum_{k=1}^{K} \alpha_k u_{n+1-k}$$

The first order BD formula is the *backward Euler* scheme

$$hu_{n+1} = u_{n+1} - u_n \qquad (\beta_0 = 1, \alpha_1 = 1)$$

It was mentioned in the previous section that stiff problems require small time steps in order to maintain stability. The stability properties of multistep methods are now considered. For each MSM a truncation error can be determined. *Absolute* or *A-stability* prevails when all the eigenvalues of the system (3.32) are negative. Thus the accumulation of past errors is prevented. Absolutely stable formulae would be the best methods for the solution of stiff problems since the region of stability extends to $-\infty$ in the left $h\lambda$ plane. Unfortunately however, as proven by Dahlquist [13], there are no explicit A-stable MSMs and the highest order implicit A-stable method is the trapezoidal rule. Thus if one is restricted to using only A-stable formulae then only low order approximations are possible and small time steps are required for sufficient accuracy. Thus A-stability satisfies the stability requirement for stiff problems but an accuracy problem remains.

Gear [22] however relaxed the A-stability requirement so that stiff components in the solution corresponding to large and negative $h\lambda$ (region $\mathcal{R}_1$) could be represented stably and non-stiff components corresponding to small and negative $h\lambda$ (region $\mathcal{R}_2$) could be represented accurately. See Figure 3.2.2. Methods possessing such properties are known as *stiffly stable* methods.

The Adams Moulton methods are not stiffly stable and hence are of no interest to us for the solution of ODE systems resulting from the semidiscretisation of parabolic problems. The BD formulae of orders 3-6 are however stiffly stable are suitable for our needs. Figure 3.2.2 shows the stability regions for the BD formulae of orders 1-6. These formulae are the presently

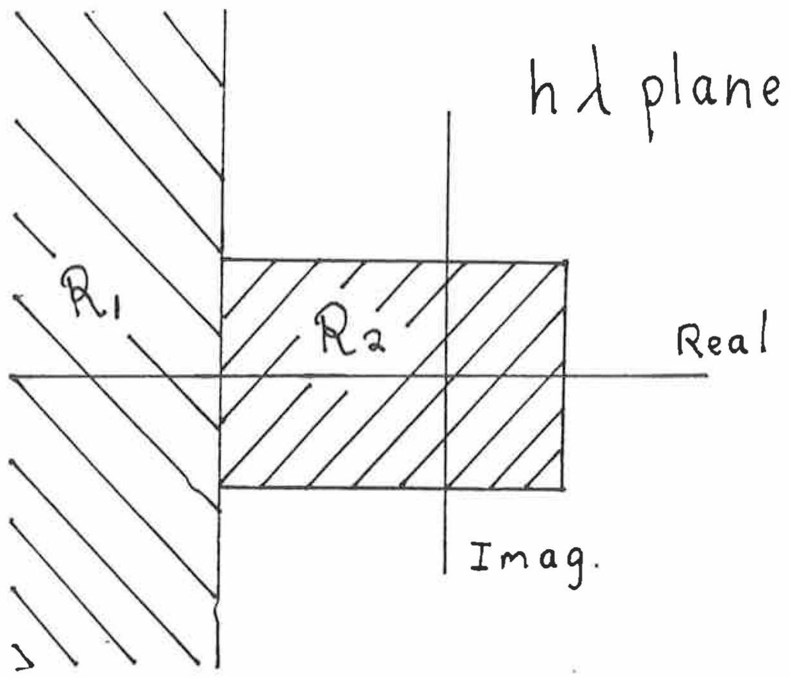


Figure 3.6: Regions of absolute stability for stiffly stable methods.

accepted standards for solving stiff systems. The pioneering software package DIFSUB [21] was written by Gear and uses BD formulae for the stiff solver option and AM methods for the non-stiff option.

Given a user specified tolerance τ the local truncation error at each step in the integration is required to satisfy $\| E_n^K \| \leq \tau$. Using analytic error expressions for each formula the maximum possible steps which might have been used in the previous step are estimated both for the present order and the nearest higher and lower formulae. The new order and step size for the next step are chosen from these estimates so that the step size is maximised. Gear in fact used a fixed step size for several steps in order to guarantee accurate error estimates. Several other heuristic approaches were also used in order to produce a robust code.

Other versions of the Gear package were later developed to take advantage of particular problem structures. For example, GEARB was developed for systems having a banded Jacobian structure and GEARIB for corresponding implicit ODE systems. The next major improvement to the Gear type ODE integrator was the implementation of fully variable stepping in the code EPISODE [26]. Presently, ODE integrators in popular use correspond

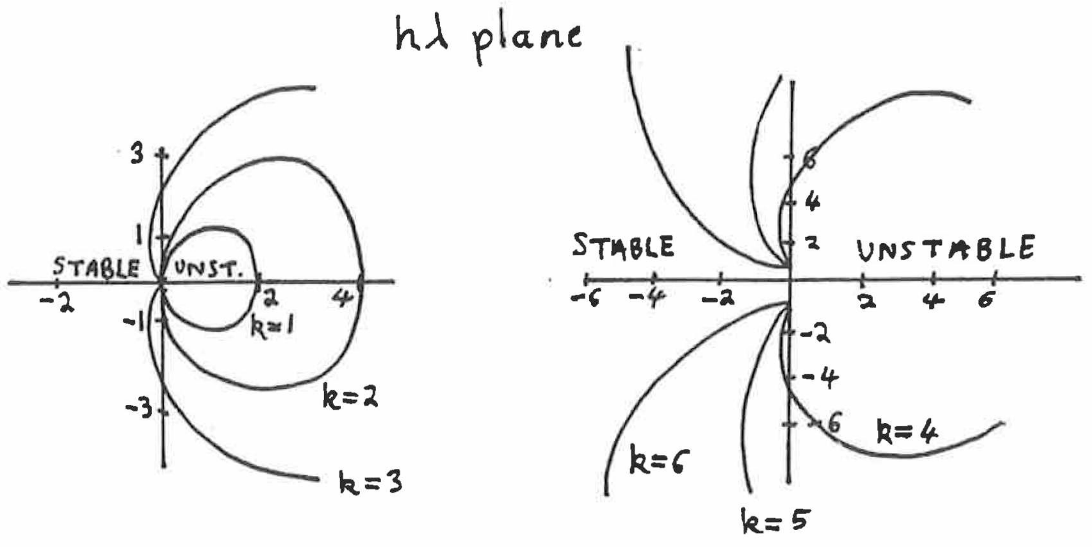


Figure 3.7: Regions of absolute stability for the BD formulae of orders 1-6.

to LSODE [24] or its variants. LSODE incorporates all the innovations to the basic Gear package outlined above. LSODI [25] is designed to solve linearly implicit ODEs and LSODIB is a variant of LSODI suitable for systems with banded Jacobians. LOSDA possesses an automatic stiffness check so that switching between stiff and non-stiff formulae may be performed automatically. This technique is based on an algorithm of Petzold [42].

Having now examined the semidiscretisation of parabolic problems and the solution of the resultant ODE system we will now examine two popular packages which implement these algorithms.

3.3 PDE Software

Presently used PDE software for parabolic equations in one space dimension incorporate the following techniques.

Automatic semidiscretisation The programs discretised the spatial interval on a user specified grid according to some generally applicable discretisation rule. The user has the responsibility of ensuring the adequacy of the spatial grid and, where available, the choice of the discretisation.

Automatic ODE solution The system of ODEs resulting from the discretisation above is solved by the program in an automatic fashion using on of the available ODE integrators described in the last section.

Details of two packages for solving parabolic type partial differential equations are now given.

3.3.1 NAG Routine D03PGF

This routine, based on the code of Sincovec and Madsen [46], is designed to solve a general system of N parabolic equations of the form

$$C_i\frac{\partial U_i}{\partial t} = \sum_{j=1}^{N}\frac{\partial}{\partial x}\left(g_{ij}\frac{\partial U_i}{\partial x}\right) + f_i \qquad i = 1, 2, \ldots, N$$

subject to the general boundary conditions

$$p_i(t)U_i + q_i(t)\frac{\partial U_i}{\partial x} = r(U_i, t) \qquad i = 1, 2, \ldots, N$$

in either Cartesian, polar or spherical polar coordinates.

The method of lines approach is used in which the spatial terms are discretised using second order centered Finite Differences. If the boundary conditions are such that q is non-zero then the order of accuracy at the boundaries is of order one. Discontinuities are permitted between the initial and boundary values and the user may choose from a limited number of fixed non-uniform grids. The ODE integrator for this package comprises the GEARIB variable order/variable step code. The initial time step is automatically chosen by the program and subsequent steps are chosen so that a user specified accuracy in the time integration is maintained. The code is robust and allows automatic resetting of the integration in the case of rejected time steps.

3.3.2 PDECOL

This routine by Madsen and Sincovec [32] solves the general system of N partial differential equations

$$\frac{\partial U_i}{\partial t} = f_i(t, x, U, U_x, U_{xx}) \quad i = 1, 2, \ldots, N$$

Since the system incorporates ODEs and the three standard types of PDEs then for each equation of the system zero, one or two boundary conditions may be needed. They must be of the form

$$b_i(U, U_x) = z_i(t)$$

and must be consistent with the initial conditions. The program semidiscretises in space using a Finite Element collocation procedure with piecewise polynomial test functions. The degree of these polynomials is required to be higher than the degree of the PDE(s) being solved. The user specifies the numerical grid and the result of the automatic semidiscretisation is the ODE system

$$A\frac{dU}{dt} = g(t, U)$$

The main restrictions in PDECOL are the requirements for continuity between the initial and boundary conditions which limits its applicability somewhat. Also the nonconservative nature of collocation methods in general makes them inappropriate for problems where a conservation law must be satisfied.

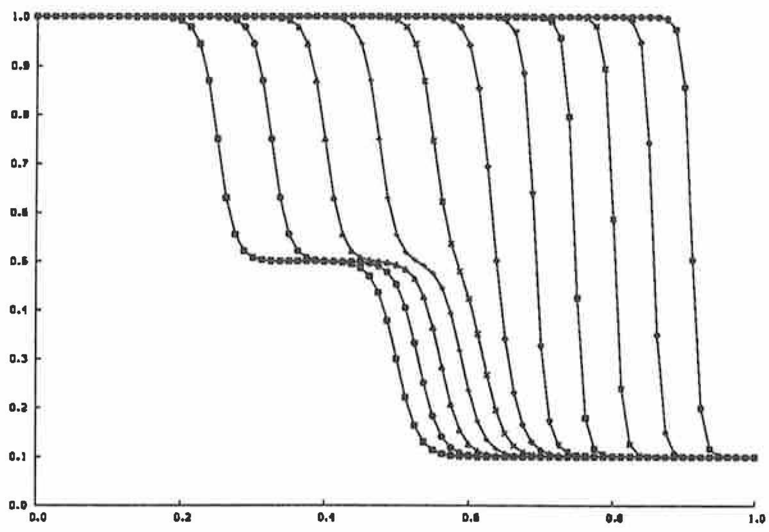


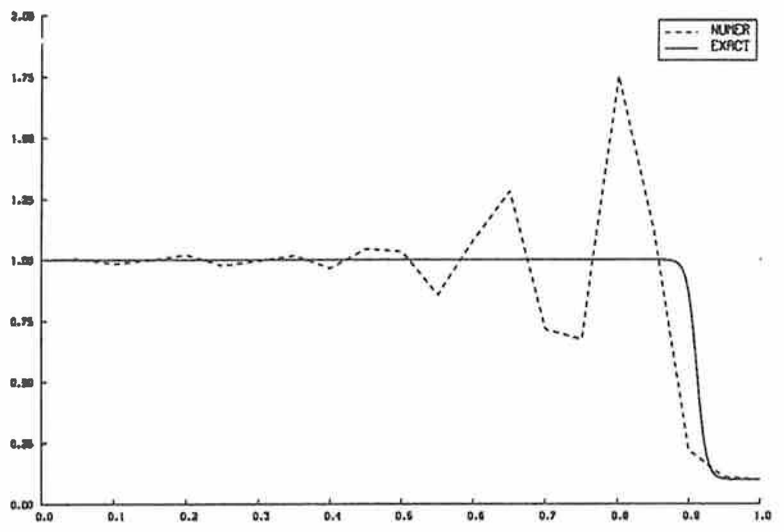
Figure 3.8: Exact solution of Burgers' equation for $\epsilon = 0.003$ at T=0(0.1)1.

3.4 Uniform grid implementations.

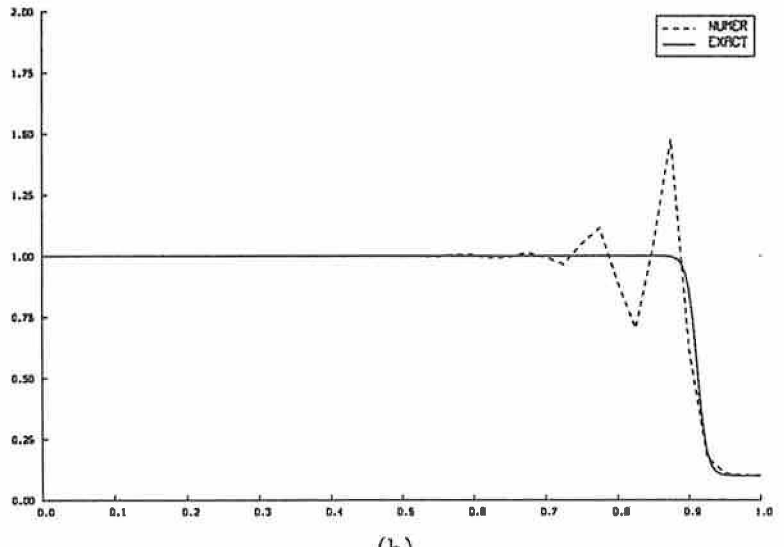
In this section we examine the results of a uniform grid solution to Burgers' equation. The NAG routine D03PGF (release 13) was used to solve this equation on the interval $T = [0, 1]$, $x = [0, 1]$ for the overtaking shocks solution (2.8) with $\epsilon = 0.003$.

The exact solution of this problem is shown in Figure 3.8. The solution exhibits both overtaking shocks and a boundary layer at x=1 during the chosen time interval and thus presents a challenging test for any program.

The solution was calculated for uniform mesh sizes of 21, 41, 81 and 161 mesh points. Figures 3.9, 3.10 and compare the numerical and analytic solutions at the final time T=1. In all but the final plot oscillations arise in the neighbourhood of the boundary layer. These oscillations are typical of Finite Difference solutions of nearly-hyperbolic equations. Only in the final example, where 161 mesh points were used, do the oscillations appear to disappear.

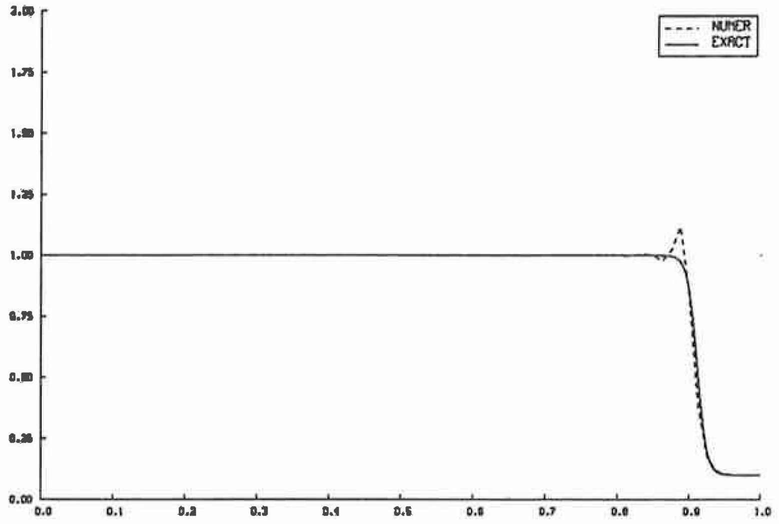


(a)

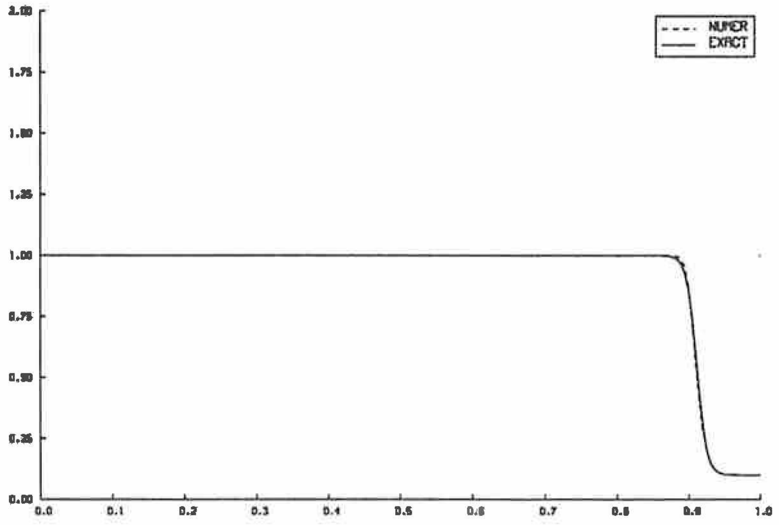


(b)

Figure 3.9: (a) Comparison of exact solution of Burgers' equation for $\epsilon = 0.003$ at T=1 with the numerical solution calculated on (a) 21 mesh points and (b) 41 mesh points.



(a)

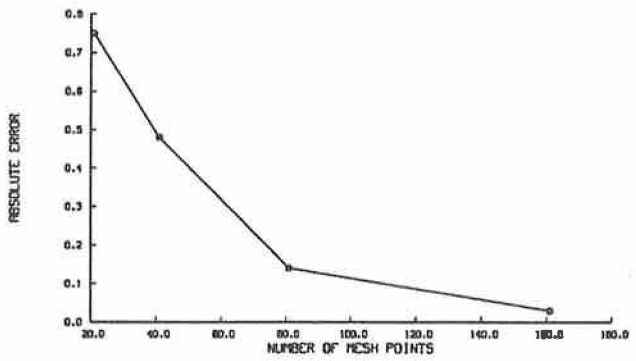


(b)

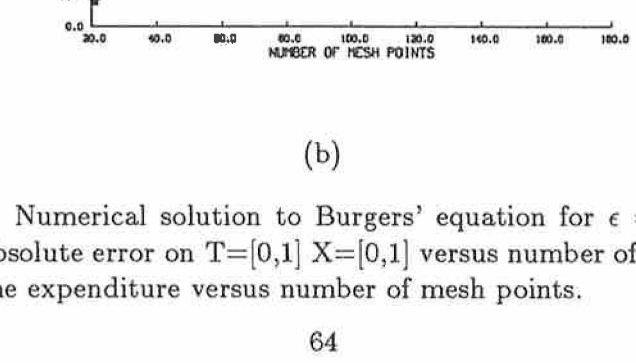
Figure 3.10: Comparison of exact solution of Burgers' equation for $\epsilon = 0.003$ at T=1 with the numerical solution calculated on (a) 81 mesh points and (b) 161 mesh points.

Both the maximum absolute error at t=1 and the CPU time were measured for each of the above implementations in order to assess their accuracy and efficiency. Figure 3.11 (a) shows the errors obtained for the four grid sizes used. The error decreases rapidly as the number of grid points increases. However even for the finest grid the maximum error is still quite large and would not be acceptable for practical use. For sufficient accuracy therefore, grids of the size of 200 or more points would appear to suffice. The effect of the grid size on the CPU time required to solve the problem is displayed in Figure 3.11 (b). The CPU time consumption appears to vary approximately linearly with the number of mesh points in this particular range of integration. However, for the finest grid, which is by no means acceptably accurate, the time is approximately 100 seconds. The actual computer used for these experiments was an APOLLO workstation.

The results shown above demonstrate clearly that for problems with rapidly propagating shocks and/or boundary layers the accuracy and efficiency of the standard uniform grid approach suffers badly. Engineering accuracy for these problems may only be obtained after considerable computer overhead and in many cases may prove to be prohibitively expensive and time consuming. Clearly more efficient methods are required for such problems. In Chapter 2 the idea of using adaptive non-uniform grids was introduced. Chapter 4 investigates this idea further and develops the theory of adaptive meshing. Using such methods the efficient solution of problems like those above is rendered possible.



(a)



(b)

Figure 3.11: Numerical solution to Burgers' equation for $\epsilon = 0.003$. (a) Maximum absolute error on T=[0,1] X=[0,1] versus number of mesh points. (b) CPU time expenditure versus number of mesh points.

Chapter 4

Adaptive Mesh Strategies

In Chapter 3 quantitative evidence of the inadequacy of uniform spatial grids in the solution of parabolic PDEs was examined. Such poor performance of uniform grids may be explained by analysing the sources of error in typical spatial discretisations. Similar discretisations may be identified, based on non-uniform grids, which offer the possibility of more accurate and efficient approximation. Given a particular solution, general non-uniform grids appear to offer superior resolution over their uniform counterparts. However, this is strictly true only if the non-uniform grid is carefully chosen.

Methods for deriving suitable non-uniform grids for particular problems are known collectively as grid generation methods. Two contrasting approaches prevail; methods where *à priori* information concerning the spatial structure of the solution is necessary and methods where such information is not required. Numerical grid generation methods are effective in the generation of grids for elliptic PDEs (time-independent problems) and in the selection of suitable initial grids for parabolic PDEs. In the latter case, an initial non-uniform grid will only resolve the evolving solution adequately, if the spatial nature of the solution changes little with time. Otherwise, an effective initial grid may prove to be useless at a later time when the spatial nature of the solution has altered significantly. Evidence from Chapter 2 suggests that parabolic equations with solutions that vary considerably with time are the rule rather than the exception. For such equations the use of a time-independent non-uniform spatial grid, perhaps the product of an effective grid generation scheme, may be less effective than the conventional uniform grid approach. A single grid generation step is not sufficient to guarantee an effective spatial mesh for parabolic-type problems. Instead, regeneration of the grid during the problem evolution is required, so that the solution is adequately resolved at all times. This is the basic approach behind adaptive meshing.

All adaptive mesh strategies are similar in that they rely upon the repeated application of particular mesh generation methods. Thus, they inherit the merits and/or deficiencies of their underlying mesh generation algorithms. Adaptive mesh strategies differ considerably, however, in the way they relate the spatial mesh to the evolving solution. On the basis of this property a two-way classification of adaptive mesh algorithms is possible, namely, **local mesh refinement methods** and **mesh moving methods**.

Important features of adaptive mesh algorithms are their robustness and generality. These are difficult to achieve in the case of parabolic problems since there exists such a wide variety. Some algorithms may require "fine tuning" to the problem at hand but this is to be avoided since it conflicts with the above two requirements. A good adaptive mesh algorithm will thus tend to be automatic and consequently easy to implement. An effective method of ascertaining the quality of a particular adaptive mesh strategy for the solution of parabolic PDEs is to couple it with existing software for such problems and monitor its performance on a wide selection of example problems.

4.1 The problem with uniform grids

In Chapter 3 section 3.4, numerical solutions to Burgers' equation were calculated, using the NAG routine D03PGF [39], for a selection of fixed uniform spatial grids. Not unexpectedly, the finer meshes require greater CPU overhead but deliver more accurate solutions. However, even after a CPU time of approximately one hundred seconds for a solution on a fine grid of 161 mesh points, a significant numerical error persists. This example demonstrates the typical problems associated with solving parabolic equations using uniform grids. Solutions may be erroneous or costly and frequently are both.

By examining typical spatial discretisations used in the solution of parabolic equations the reasons for such deficiencies may be determined. Consider the standard spatial discretisations based on the uniform mesh

$$\Gamma_N : \varsigma_i = i * h \quad i = 1, 2, \ldots, N+1$$

where

$$h = \frac{(b-a)}{N}$$

In Chapter 3 the following Finite Difference replacements of y_i' and y_i'' were derived.

$$y_i' = \frac{y_{i+1} - y_{i-1}}{2h} + E_1 \tag{4.1}$$

$$y_i'' = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + E_2 \tag{4.2}$$

The leading terms of the truncation errors E_1 and E_2 are, respectively,

$$E_1 = -\frac{h^2}{6} y^{(iii)} \tag{4.3}$$

$$E_2 = -\frac{h^2}{12} y^{(iv)} \tag{4.4}$$

In both cases although the truncation error is proportional to h^2 it is also proportional to higher derivatives of the solution. These higher derivatives will be significant where the solution varies rapidly in space, resulting in relatively large truncation errors in such areas. This is the reason why large errors were observed in the numerical experiments of section 3.4. In general, the use of uniform grids means that the truncation error will be non-uniformly distributed over the spatial domain.

The simplest way of ensuring sufficiently small truncation errors over the entire spatial domain is to use a uniform but fine spatial grid. This allows the maximum truncation error (corresponding to the region of greatest spatial variation) to be limited, since the factor of h^2 in the truncation errors (4.3) and (4.4) will be considerably reduced. In doing this, however, the grid spacing will also be reduced in regions where the truncation error is already sufficiently small. This approach, therefore, over-compensates in that it reduces the spatial errors globally, rather than locally, where necessary. The overall result is that calculation time is unnecessarily increased as demonstrated by the example results in section 3.4.

Since the truncation errors in (4.3) and (4.4) are proportional to both h^2 and higher solution derivatives, it is possible to uniformly distribute these errors over the spatial domain by selecting relatively small grid spacings where the derivatives are large and relatively large grid spacings elsewhere. This suggests the use of truly non-uniform grids in space.

In order to assess the viability of non-uniform spatial grids, the standard discretisations based on such grids must be examined. Consider the interval [a,b] divided into N mesh spacings by the non-uniform grid

$$\Pi_N : a = x_0 < x_1 < x_2 < \ldots < x_N = b$$

Choosing the notation

$$p = \Delta x_{i+1} = x_{i+1} - x_i \qquad q = \Delta x_i = x_i - x_{i-1}$$

the Finite Difference replacements of y_i' and y_i'' based on the non-uniform grid Π_N may be written as follows

$$y_i' = \frac{1}{p+q}\left\{\frac{q}{p}(y_{i+1} - y_i) + \frac{p}{q}(y_i - y_{i-1})\right\} + E_1 \qquad (4.5)$$

$$y_i'' = \frac{2}{p+q}\left\{\frac{1}{p}(y_{i+1} - y_i) - \frac{1}{q}(y_i - y_{i-1})\right\} + E_2 \qquad (4.6)$$

where

$$\begin{aligned} E_1 &= -\frac{pq}{6}y^{(iii)} - \frac{pq}{24}(p-q)y^{(iv)} \\ &\quad -\frac{pq}{120}(p^2 - pq + q^2)y^{(v)} + \dots \end{aligned} \qquad (4.7)$$

$$\begin{aligned} E_2 &= -\frac{p-q}{3}y^{(iii)} - \frac{1}{12}(p^2 - pq + q^2)y^{(iv)} \\ &\quad -\frac{1}{60}(p^3 - p^2q + pq^2 - q^3)y^{(v)} + \dots \end{aligned} \qquad (4.8)$$

These formulae were derived in Chapter 3 and degenerate to those of (4.1) and (4.2) in the case of a uniform grid (p = q = h).

For the approximation of y_i' the leading term of the truncation error is

$$-\frac{pq}{6}y^{(iii)}$$

Thus, the error remains second order in terms of p and q when a uniform grid is replaced by a non-uniform grid. However, in the case of y_i'', the truncation error degenerates to first order since its leading term becomes

$$-\frac{p-q}{3}y^{(iii)}$$

The degeneration of the formal truncation error, in the approximation of y_i'', has important consequences for the numerical solution of parabolic equations, since they are characterised by the presence of such a spatial derivative term. In general, the use of non-uniform spatial grids in the Finite Difference discretisation of parabolic problems, leads to a reduction of one in the order of accuracy of the method. This would appear to severely limit the applicability of non-uniform spatial grids to such problems. However, by an appropriate restriction on the choice of non-uniform grid, formal second order accuracy may be restored. The leading term of the truncation error, (4.1), can be made second order in terms of p and q if the following restriction on their relative sizes is maintained.

$$\begin{aligned}(p-q) &= \kappa q^2 \\ p &= q(1+\kappa q) \qquad (4.9) \\ \kappa = \mathrm{O}(1) & \qquad \text{for i=1,2, \ldots ,N}\end{aligned}$$

If the non-uniform mesh is chosen so that (4.9) holds, second order accuracy will be possible in the approximation of y_i' and y_i''. In the next section, on mesh generation, a method for generating a non-uniform grid using the relation (4.9) (called the κ method by Noye [40]), will be described.

In Chapter 3 the non-uniform mesh in x, Π_N, was visualised as a transformation $x(\varsigma)$ from the uniform grid in ς, Γ_N. This allows a straightforward means of determining truncation errors. For example, the errors for the approximations of y_i' in (3.6) and y_i'' in (3.10) were

$$E_1 = -\frac{1}{6}H^2 x_\varsigma^2 y_i^{(iii)} + O(H^4) \qquad (4.10)$$

$$E_2 = -\frac{H^2}{12}\{4x_{\varsigma\varsigma} y_i^{(iii)} + x_\varsigma^2 y_i^{(iv)}\} + O(H^4) \qquad (4.11)$$

where H is the uniform grid size in the ς coordinate and x_ς, $x_{\varsigma\varsigma}$ are the local gradient and curvature of the mesh transformation.

In terms of H both errors are of the same order. The difficulty mentioned in Chapter 3 concerning the accurate representation of y_i'' is explained by the rather complex terms appearing in E_2 above. Maintenance of second order accuracy in terms of H^2 requires that

$$4x_{\varsigma\varsigma}y_i^{(iii)} + x_\varsigma^2 y_i^{(iv)} = O(1)$$

This restriction is equivalent to that of (4.9) but shows more clearly the influence of the higher y derivatives on the truncation error. Second order accuracy can be maintained by ensuring that the derivatives of the mesh transformation are small when the corresponding y derivatives are large. This of course relies on the continuity and boundedness of the y derivatives. A further consequence of the use of non-uniform grids is their effect on the stability of the numerical methods used in the time integration. For explicit methods the stability depends on the smallest mesh spacing. See Mitchell and Griffiths [37] for a stability analysis of some common Finite Difference schemes. For small mesh spacings, the time step must be reduced so that the method remains stable. In such cases the considerations of spatial resolution and stability may conflict and the superior resolution gained by reducing the mesh spacing may be more than offset by the increased number of time steps needed in order to maintain stability. The use of explicit methods in conjunction with non-uniform grids is by no means inappropriate however, as long as some control over the minimum mesh spacing is exercised. Madsen [32] addresses this particular problem in his presentation of a non-stiff adaptive meshing technique, to be discussed in the following section.

As demonstrated in 3.2, the semidiscretisation of parabolic problems leads to systems of ODEs which are stiff. The methods of integration for such systems must be implicit due to the stringent stability requirements that prevail. Therefore the effect of variable grid spacing on the time integration is not of major importance since available ODE integrators for stiff systems incorporate implicit formulae. However in the adaptive meshing techniques to be discussed, excessive stiffness is limited by preventing the nodes from becoming too close.

We will now consider the techniques for generating improved meshes based on non-uniform grids. These techniques are collectively known as numerical grid generation methods.

4.2 Numerical grid generation

Numerical grid generation concerns the selection of appropriate non-uniform grids for particular problems. The choice of grid is entirely dependent on the nature of the problem and some grid generation methods therefore require specific à priori knowledge of the problem being solved. Methods which are less dependent on such information are robust and thus more generally applicable. As already pointed out, this is important when dealing with parabolic PDEs. In the following sections several common approaches to grid generation are described.

4.2.1 The κ-method

In 4.1 the effect of non-uniform grid spacing on the Finite Difference representation of y_i'' was investigated. It was seen that, for parabolic problems in general, the order of accuracy attainable on a non-uniform grid tends to be of order one less than that obtainable on a uniform grid. However, a non-uniform grid denoted by

$$\Pi_N : a = x_0 < x_1 < x_2 < \ldots < x_N = b$$

can be made to retain second order accuracy if the following restriction holds.

$$p = q(1 + \kappa q) \qquad \kappa = O(1) \qquad i = 1, 2, \ldots, N$$

Different values of κ correspond to different grid configurations that are smoothly varying. For example, a positive value of κ produces a grid which starts with a small spacing and increases monotonically to larger values. If κ is negative then the reverse occurs. Figure 4.1 shows examples of grids generated in this fashion for various values of κ. Grids produced by the κ

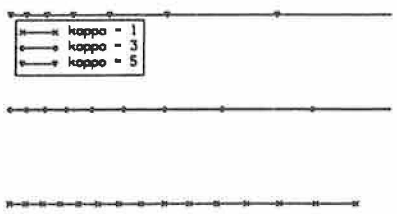


Figure 4.1: Grids generated by the κ method for κ=1,3,5.

method tend to retain their accuracy. However the method does not take into account the behaviour of the solution which, as was seen in section 4.1, plays an important role in the determination of the overall truncation error.

The κ method is therefore useful if the mesh configuration (determined by the value of κ) is known in advance. This might be the case, for instance, in a problem possessing a known boundary layer at x=0. Choosing $\kappa \geq 5$ would generate a mesh suitable for resolving such a feature in the solution. The κ method therefore relies completely on à priori knowledge of the problem being solved. Not unexpectedly, this limits the applicability of the method to well known problems with predictable solutions. Unfortunately, the very problems which would benefit the most from the use of non-uniform spatial grids are those problems which exhibit unpredictable solution structures such as shocks and boundary layers.

4.2.2 Mapping functions

In Chapter 3 the notion of a non-uniform grid as a transformation of a corresponding uniform grid was a useful one. Given a uniform mesh on the spatial interval [0,1]

$$\Gamma_N : \varsigma_i = i * h \qquad h = \frac{1}{N} \qquad i = 0, 1, 2, \ldots, N$$

and a corresponding non-uniform mesh

$$\Pi_N : 0 = x_0 < x_1 < x_2 < \ldots < x_N = 1$$

The non-uniform mesh may be regarded as the result of applying the mapping function $x(\varsigma)$ to the uniform mesh.

The mapping function approach uses an appropriate function which serves to generate such a non-uniform mesh. Choosing the right mapping function allows the structure of the non-uniform grid to be controlled. Obviously the function will have to be monotonically increasing/decreasing so that the resultant non-uniform grid is contiguous. An example of such a function is

$$x(\varsigma_i) = \sin\left(\frac{\varsigma_i \pi}{2}\right) \qquad i = 0, 1, \ldots, N$$

This maps a uniform grid in ς, on the interval [0,1], onto a non-uniform grid in x, on the same interval. For other intervals, the resulting grid can be scaled by multiplying the mapping function by an appropriate constant. The non-uniform grid is produced by projection from a circle as shown in Figure 4.2. In this grid mesh points are concentrated at x=1. As in the κ method a knowledge of the solution is required for a successful grid generation to be possible. A typical use of the method would be the solution of a problem possessing a boundary layer whose location is known.

4.2.3 Stretched coordinates

In the mapping function approach discussed above, a uniform mesh in some coordinate ς was used to generate a non-uniform mesh in the spatial coordinate x. The problem being solved was then discretised on the non-uniform

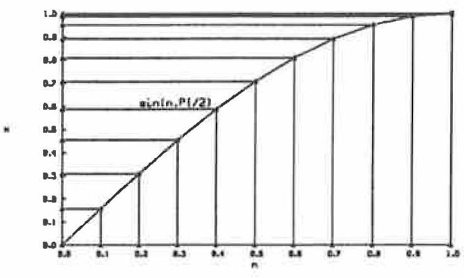


Figure 4.2: Non-linear mesh generated by the mapping function $x(\varsigma) = \sin\left(\frac{\varsigma\Pi}{2}\right)$

mesh in x.

The use of *stretched* coordinates is effectively the converse of the mapping function idea. Instead of discretising the problem on a non-uniform grid in x the original differential equation is transformed (using a function) to a new coordinate ς. The "stretching" function is chosen so that large gradients in the solution in the x coordinate are stretched out in the new coordinate. This allows a uniform mesh to be used for the discretisation of the transformed equation. Coordinate stretching therefore avoids the deterioration in formal truncation error when an equation is discretised on a non-uniform spatial mesh.

Vinokur [55] analyses several types of one-dimensional stretching functions for use in Finite Difference calculations. The two main types of functions considered are *interior* and *two-sided* stretching functions. Functions of these types are based respectively on the inverse hyperbolic sine and tangent functions.

The main disadvantage of stretching functions is that new terms may appear in the transformed equation which may need to be considered when choosing an appropriate spatial approximation. For example, Braddock and Noye [4] introduce a general stretching function for the solution of the simple heat equation

$$\frac{\partial U}{\partial t} = D\frac{\partial^2 U}{\partial x^2}$$

The spatial coordinate x was rescaled by the transformation $x = H(\varsigma)$ such that an equispaced mesh in ς corresponded to a non-uniform mesh in x. With this transformation the equation transforms as follows.

$$\begin{aligned} \frac{\partial U}{\partial x} &= \frac{\partial U}{\partial \varsigma}\frac{\partial \varsigma}{\partial x} = \frac{\frac{\partial U}{\partial \varsigma}}{\frac{\partial \varsigma}{\partial x}} \\ \frac{\partial^2 U}{\partial x^2} &= \frac{\frac{\partial H}{\partial \varsigma}\frac{\partial^2 U}{\partial \varsigma^2}\frac{\partial \varsigma}{\partial x} - \frac{\partial U}{\partial \varsigma}\frac{\partial^2 H}{\partial \varsigma^2}\frac{\partial \varsigma}{\partial x}}{\frac{\partial H}{\partial \varsigma}^2} \end{aligned} \tag{4.12}$$

Note

$$\frac{\partial \varsigma}{\partial x} = \frac{\partial H}{\partial \varsigma} = H'(\varsigma)$$

Thus, the equation becomes

$$\frac{\partial U}{\partial t} = \frac{D}{(H'(\varsigma))^2}\frac{\partial^2 U}{\partial \varsigma^2} - \frac{D}{H''(\varsigma)}(H'(\varsigma))^3\frac{\partial U}{\partial \varsigma}$$

This is a convection-diffusion equation with variable coefficients. The function $H(\varsigma)$ is selected so that $H'(\varsigma)$ is small in regions where the solution changes rapidly. In solving (4.2.3) numerical diffusion may be introduced by the discretisation of the convective term, as discussed in 3.1.1. In this case the solution of the transformed equation on a uniform mesh may present more difficulties than the original equation. However, the extra work needed to solve the transformed equation may be compensated for by the improved resolution in the transformed coordinate.

The situation above involving the arbitrary transformation $x = H(\varsigma)$ is similar to that of the mapping function approach in that an appropriate transformation must be selected prior to solving the problem. This relies once again on sufficient à priori knowledge of the solution behaviour.
White [57] considers a solution-dependent coordinate transformation for initial/boundary-value problems of the form

$$A(y,x,t)y_t + B(y,x,t)y_x = C(y,x,t)$$

based on the solution arc length.

From integral calculus the arc length of a function y(x) is

$$ds = \sqrt{1 + \left(\frac{dy}{dx}\right)^2}\, dx$$

Thus

$$\begin{aligned} \left(\frac{ds}{dx}\right)^2 &= 1 + \left(\frac{dy}{dx}\right)^2 \\ 1 &= \left(\frac{dx}{ds}\right)^2 + \left(\frac{dy}{ds}\right)^2 \\ 1 &= x_s^2 + y_s^2 \end{aligned} \tag{4.13}$$

where subscripts imply partial differentiation. Equation (4.13) suggests that arc length is a suitable choice for a coordinate transformation since the derivatives x_s and y_s are bounded by 1 regardless of how large $\frac{dy}{dx}$ becomes. In approaches based on arc length the contributions of x and y to the total solution arc length are presumed to be about the same.

White considers the transformation from coordinates (x,t) to new coordinates (s,T) where s is an arc length coordinate given by

$$s = \int_0^x \left(1 + \left(\frac{dy}{dp}\right)^2\right)^{\frac{1}{2}} dp/\theta$$

where

$$\theta = \int_0^1 \left(1 + \left(\frac{dy}{dp}\right)^2\right)^{\frac{1}{2}} dp$$

and

$$T = t$$

θ normalises the the total arc length of the solution over the spatial interval [0,1]. This leads to a slightly more general form of (4.13)

$$\theta^2 = x_s^2 + y_s^2$$

The use of an arc length transformation introduces nonlinearities into the existing equation to be solved and augments this with an extra differential equation which defines the arc length transformation x(s) back to the original coordinates (x,t). Since the solution in arc length coordinates y(s) no longer has large derivatives (as a consequence of (4.13)) a uniform mesh may be used to discretise the problem in (s,T) coordinates. In the original (x,t) coordinates the effect of the transformation is to place nodes at equal intervals of arc length along the solution curve and so clustering nodes in regions of large solution gradients. The transformation therefore provides automatic mesh selection. White's method is an improvement on the previously described stretching function approach of Noye in that the choice of function is based solely on the current solution values and not on other à priori information about the character of the solution. This makes White's method potentially very robust.

One possible difficulty with the arc length approach is that the transformation will be less reliable if a large change in the direction of the solution curve occurs over a short distance. Ablow and Schechter [2], in their investigation of stretched coordinates in the solution of boundary-value problems in ordinary differential equations, attempt to remedy this situation by including a dependence on the angular inclination of the solution curve in the transformation. They use an alternative transformation t which is a linear combination of both arc length s and angular variation ω.

$$t = s + C \int \mid d\omega \mid$$

where ω is the angular inclination of the solution and C is a constant length. The resulting transformation

$$(1 + C \mid \omega_s \mid)^2 \left(x_t^2 + y_t^2\right) = 1 \tag{4.14}$$

Ablow and Schechter show that increasing the contribution of the curvature in the definition of the coordinate transformation (by increasing C) causes a relative increase in truncation error in straight regions of the solution and a relative decrease in truncation error in curved regions. As in the case of White's arc length transformation this transformation complicates the differential system to be solved. The choice of C, however, is not obvious and makes the implementation less robust.

4.2.4 Equidistribution

In an equidistribution strategy the grid points are placed so that some positive weight function is equally distributed over the spatial interval. This requires that the following condition holds at each mesh interval,

$$\int_{x_i}^{x_{i+1}} w(x)dx = C \qquad (4.15)$$

where w(x) is a chosen weight function and C is a constant. The discrete form of (4.15) is

$$h_i w_i = C$$

Using this strategy the mesh spacings will be small where w(x) is large and vice versa.

The non-uniform grid distribution in x may be interpreted as a transformation $x(\varsigma)$ of a uniform grid in some coordinate ς. If successive integer values of ς are chosen to define the uniform grid then $\Delta\varsigma = 1$ and h_i becomes $\Delta x/\Delta\varsigma = x_\varsigma$. Equation (4.15) becomes

$$x_\varsigma w = C \qquad (4.16)$$

The grid distribution resulting from such an equidistribution principle may be interpreted in a variational sense. Two possible interpretations of (4.16) arise depending on whether w is a function of x or ς. If w is a function of ς (the points themselves) then (4.16) becomes

$$x_\varsigma w(\varsigma) = C$$

This is in fact the Euler equation [1] for the minimisation of the integral

$$\int_0^1 w(\varsigma) x_\varsigma^2 d\varsigma \tag{4.17}$$

Equation (4.17) can be visualised as representing the energy of a system of springs with spring constants $w(\varsigma)$. The equidistribution relation therefore corresponds to the state of minimum energy (equilibrium) of this spring system.

If the weight function in (4.16) is taken to be a function of x (the point locations) then the equation becomes

$$x_\varsigma w(x) = C$$

This is the Euler equation corresponding to the integral

$$\int_0^1 [w(x) x_\varsigma]^2 d\varsigma \tag{4.18}$$

This integral can be interpreted as the least squares minimisation of the cumulative grid spacings weighted by the function w(x). In this case the equidistribution principle leads to the smoothest possible grid distribution.

[1]The function $x(\varsigma)$ for which the integral $\int_0^1 F(\varsigma, x, x_\varsigma) d\varsigma$ is an extremum is given by the solution to Euler differential equation

$$\frac{d}{d\varsigma}\left(\frac{\partial F}{\partial x_\varsigma}\right) - \frac{\partial F}{\partial x} = 0$$

Both the spring analogy and the smoothness approach are valid for a particular equidistribution strategy. However the smoothness approach (involving w(x)) is the most commonly used since it is expedient to associate the weight function with some physical property which varies in space.

The weight function w(x) is usually a direct function of the solution, for example the first derivative. Thompson [50] indicates that if w(x) is some measure of the solution error then equidistributing w(x) is asymptotically optimal and this optimal error is stable under perturbations of the grid distribution. For this reason he suggests that it is unnecessary to locate the points with excessive accuracy. The constant in equation (4.15) can be determined by normalising the weight function over the spatial interval [0,1]. If the spatial interval is divided into N mesh intervals then (4.15) becomes

$$\int_{x_i}^{x_{i+1}} w(x)dx = \frac{1}{N}\int_0^1 w(x)dx \qquad (4.19)$$

Various choices of weight function are possible. The best choice, however, is the truncation error itself since the ultimate aim of the grid generation is to equidistribute this quantity over the spatial domain. Davis and Flaherty [14] equidistribute a weight function related to the truncation error of the Finite Element method they use. One of the main disadvantages of using a truncation error estimate as the weight function is the necessity for estimating higher solution derivatives. For instance, in the method of Davis and Flaherty above, estimates of the second solution derivative are required if linear trial functions are used. If cubic trial functions are used then estimates of the fourth solution derivative are necessary. Usually such estimates are subject to considerable computational noise which renders their effectiveness questionable.

The alternative approach is to derive weight functions based on lower derivatives of the solution. Two common choices are

$$w = U_x \qquad (4.20)$$

$$w = \sqrt{1 + U_x^2} \tag{4.21}$$

The second choice is based on the arc length of the solution curve. White's method, discussed in 4.2.3 makes use of a coordinate transformation of the form

$$s = \int_0^x \sqrt{1 + y_x^2} dx / \theta$$

where

$$\theta = \int_0^1 \sqrt{1 + y_x^2} dx$$

This transformation corresponds to the equidistribution relation (4.19) above where the weight function is $w(x) = \sqrt{1 + U_x^2}$. The approach of White can therefore be regarded as equidistribution of solution arc length by transformation.

Equidistributing meshes generated by the criteria (4.20) and (4.21) above are shown in Figures 4.3 and 4.4, respectively.

For the first mesh function the mesh spacing becomes infinite in regions where the solution becomes flat ($u_x = 0$). This is because the equidistribution problem, equation (4.15), has no unique solution when $w(x) \rightarrow 0$. Numerical difficulties in such cases can be overcome by including a constant regularising term in the definition of the weight function. Such a term appears in the second weight function above. Figure 4.4 shows that as the slope of the solution approaches zero the mesh spacing becomes uniform.

Weight functions such as (4.20) and (4.21) based solely on the first derivative tend to treat solution extrema (where $u_x = 0$) in the same manner as

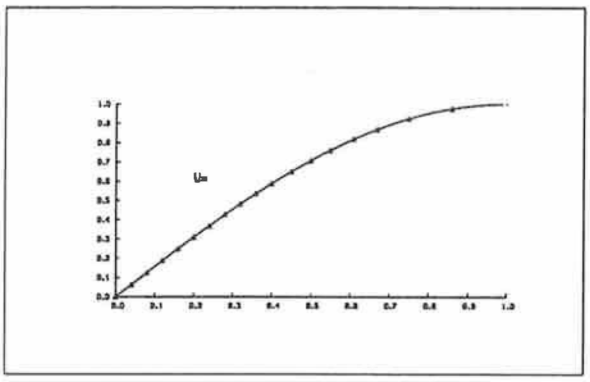


Figure 4.3: Equidistributing mesh generated using the weight function $w = U_x$.

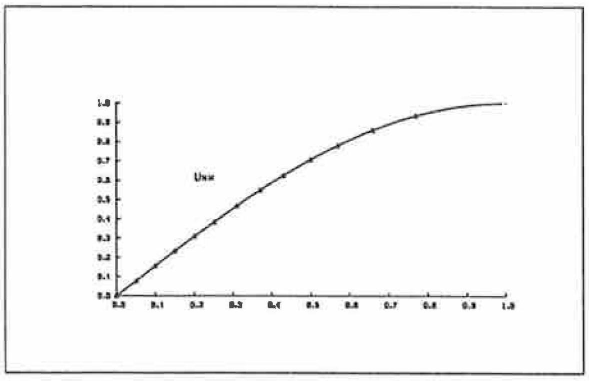


Figure 4.4: Equidistributing mesh generated using the weight function $w = (1 + U_x^2)^{1/2}$.

flat regions. However the solution changes rapidly at such points and the relatively large value of the curvature will augment the truncation error. Equidistribution based on the second derivative u_{xx} may also be used. For instance, Blom et al. [3] present an adaptive moving grid method in which the weight function is

$$w = \sqrt{\alpha + \mid U_{xx} \mid} \tag{4.22}$$

where α (=1 usually) regularises the weight function in cases where $U_{xx} = 0$. Coyle et al. [11] also consider weight functions of the form (4.22) in which α is 0 and 1.

Just as in the case of first derivative equidistribution, mesh functions based solely on the second derivative tend to be biased towards concentrating the mesh points at solution extrema. A combination of the two types of approach is possible if the following weight function is used.

$$w = 1 + \alpha^2 \left| \frac{U_{xx}}{(1 + U_x^2)^{\frac{3}{2}}} \right| \tag{4.23}$$

This function causes points to be concentrated near solution extrema. Regions of zero curvature are assigned a uniform mesh spacing.

Recalling the equidistribution equation (4.15) the "equals" could conceivably be replaced by "less than or equals". A mesh generated by this equation would constitute a ***sub-equidistributed*** mesh. The work required to calculate a sub-equidistributed mesh is usually less than that required for an equidistributed mesh and is usually as effective even though it overkills somewhat. For example, Smooke and Koszykowski [48] sub-equidistribute the difference between solution components and derivatives using the following replacements of (4.15).

$$\int_{x_i}^{x_{i+1}} \frac{dy}{dx} dx \leq \delta \mid \max y_i - \min y_i \mid$$

$$\int_{x_i}^{x_{i+1}} \frac{d^2y}{dx^2} dx \leq \gamma \mid \max \frac{dy_i}{dx} - \min \frac{dy_i}{dx} \mid$$

where γ and δ are small and less than 1. Madsen [32] describes a sub-equidistributing mesh strategy where the weight function is chosen to be a weighted average of five different functions of the solution.

4.3 Adaptive meshing

The use of a single non-uniform grid is appropriate only in the solution of problems which are time-independent, for example, elliptic problems such as the Laplace equation, or time-dependent problems, where the regions of spatial activity in the solution remain fixed as time evolves, for example, Problem 5 of Chapter 2. For parabolic type problems the tendency is for solutions whose nature varies strongly with time. This is apparent from the example problems described in Chapter 2. Once-off grid generation is therefore of little use and may, as in the case of the solution to Problem 4 above, be less effective than the uniform mesh approach. The need to maintain the suitability of the mesh as the problem evolves must be addressed. This is the area of adaptive spatial meshing. In principle the methods of grid generation of the last section may be used repeatedly throughout the time evolution of the problem rather than just at the initial stage. At any time in the problem evolution the current solution may be regarded as the initial conditions for the remainder of the time integration. However, once the integration has begun the approximate solution may only be available in discrete form whereas the initial conditions may be available in analytic form, free of errors. This is of little importance in grid generation methods which rely solely upon information concerning the present discrete form of the solution.

Once a grid configuration is altered, a need to transfer information from the old to the new grids exists. Adaptive mesh algorithms differ mainly in the way they approach this problem. The following two-way classification of adaptive mesh strategies is possible on the basis of how the evolving solution and the grid are related.

- **Local mesh refinement.** Here the spatial mesh adapts with the evolving solution in response to local solution characteristics. Typically, the spatial mesh remains fixed for intervals of time after which mesh regeneration takes place at discrete time levels. The number of mesh points is variable and, in principle, local mesh refinement methods can resolve arbitrarily small scale structures in the evolving solution.

- **Mesh moving methods.** Here the spatial mesh adapts continuously with the evolving solution in a mutually-dependent manner. Usually a fixed number of grid points are moved simultaneously so that the solution is resolved as uniformly as possible over the spatial domain.

In the following section a representative selection of both types of adaptive mesh techniques are described.

4.3.1 Local mesh refinement methods

In this approach to adaptive gridding, steep gradients in the solution are resolved by inserting/deleting spatial grid points throughout the integration. The solution of the problem and the determination of the grid are treated as separate tasks. The solution is monitored throughout the integration so that mesh refinement can be implemented when appropriate. When a refinement is performed, a transfer of the solution from the old grid to the new is carried out. In the case of some Finite Element methods where the solution at each time level is represented analytically, for example collocation, there is no problem expressing the solution on the new grid. For Finite Difference methods however, the solution at each time step is only available in discrete form and thus the transfer of information from the old grid to the new requires interpolation. In such cases the interpolation method must be as accurate as the semidiscretisation so that the order of accuracy of the semidiscretisation truncation error is maintained.

A typical local mesh refinement algorithm is that of Chong [8]. The idea here is to locate the position of the shock and track its movement (if any) during the evolution of the problem. By considering such stretching functions Chong [8], in his adaptive grid scheme, chooses a mapping function solely from analysis of the solution gradient. The complete reliance on specific à priori knowledge of the solution behaviour is thus avoided. Chong's method is however restricted to a certain class of problems the solutions of which satisfy the following conditions.

- There exists a single boundary layer in the spatial interval $a \leq x \leq b$ of thickness $O(\epsilon)$.
- Inside the layer the first four derivatives of the solution are

$$\frac{d^n U}{dx^n} = O\left(\epsilon^{-n}\right) \qquad n = 1, 2, \ldots, 4$$

- Outside this region the derivatives are

$$\frac{d^n U}{dx^n} = O\left(1\right) \qquad n = 1, 2, \ldots, 4$$

By estimating the position and thickness of the boundary layer in the solution a non-uniform grid is generated by choosing a mapping function ψ which gives a linear transformation within the layer

$$\psi = \alpha\epsilon x \quad \textit{where} \quad \alpha = O(1)$$

Here ϵ is the boundary layer thickness. In the transformed coordinate the boundary layer is stretched out by a factor of $\alpha\epsilon$. Outside this region the mesh is gradually increased in size according to the following restriction

$$h_i = h_{i-1}(1 - \kappa h_i) \qquad \kappa = O(1)$$

This is the identical approach as used in the κ method of the previous section and guarantees the maintenance of second order accuracy for small values of κ. Given the restrictions imposed in the above list, Chong proves that a non-uniform mesh can retain second order accuracy when conventional Finite Differences are used to semidiscretise the problem.

The adaptive algorithm begins with estimating the boundary layer position and thickness. This is done by iteratively scanning and refining the initial grid so that the position of the layer is found accurately. At each refinement the solution is transferred from the old grid to the new by interpolation. The centre of the layer is taken as the mesh point/interval corresponding to the maximum absolute value of the solution first derivative U_x. The reciprocal of this value is interpreted as the shock thickness.

The adaptive grid strategy monitors the solution throughout its evolution in time so that the position and thickness of the shock are always known. Based on heuristic approaches the appropriate times at which the mesh should be refined are determined. Usually this is done when the shock position has altered by a prefixed amount. Determining a new grid is done in the same way as for the initial conditions. Iteration however, is not necessary since the new and old grids will not be very different if the shock is not allowed to propagate too far between successive refinements. Following a grid refinement, the transfer of the solution to the new grid is performed using interpolation. Chong finds cubic splines to be quite effective since they are second order accurate and therefore match the accuracy of conventional Finite Differences. He reports that the extra CPU time necessary to solve the tridiagonal linear system, associated with the cubic spline calculation, is not a significant part of the overall CPU usage for the algorithm.

Chong's approach, however, appears to dwell very much on heuristics. For instance, he suggests that the variable mesh can be made less sensitive to small shifts in the centre of the shock by placing 50 mesh intervals of minimum size (defined by the mesh size at the centre of the shock) in the regions adjacent to the shock centre. For problems with moving shocks this algorithm might therefore prove inefficient. Indeed Chong appears to anticipate such difficulties and, for problems involving rapidly propagating shocks over long time intervals, he recommends a preliminary transformation of the equation to a new coordinate where the shock motion is frozen or slowed down. Such a transformation can only be effected when sufficient à priori knowledge of the shock trajectory is available (usually from an inviscid analysis). Chong attempts such a transformation for Burgers' equation but only succeeds in slowing down the shock in the transformed coordinate.

The approaches of White [57] and Ablow and Schechter [2] involve solution-dependent coordinate transformations. The two approaches are similar in that they both introduce nonlinearities into originally linear equations. However, à priori knowledge of the solution behaviour is not needed and no restrictions on the form of the solution apply as in the case of Chong's mapping function approach. In fact, White criticises Chong's method on the basis that the restrictions imposed on the solution by the latter are not generally satisfied by classical boundary layers. White's method is restricted to first order PDE systems and parabolic equations must therefore be expressed as an equivalent first order system. Expressing Burgers' equation in such a form and applying Keller's box scheme [28], White demonstrates effective results for parabolic type problems. The main disadvantage of White's method is that a parabolic problem is reduced (on semidiscretisation) to a system of differential algebraic equations DAEs. This class of problem is significantly more difficult to solve than the ODE problems resulting from more conventional discretisations.

White [57] derives a more general adaptive mesh strategy using a transformation which equidistributes the arc length of the solution. The number of grid points is constant and the solution is determined on a uniform grid in arc length coordinates. Smooke and Koszykowski [48] construct an adaptive strategy by sub-equidistributing a positive weight function of the form

$$\int_{x_i}^{x_{i+1}} \frac{dU}{dx} dx \quad \leq \quad \partial \mid \max U_i - \min U_i \mid \tag{4.24}$$

$$\int_{x_i}^{x_{i+1}} \frac{d^2U}{dx^2}dx \quad \leq \quad \gamma \mid \max \frac{dU_i}{dx} - \min \frac{dU_i}{dx} \mid \tag{4.25}$$

where γ and ∂ are small and less than 1 and the maximum and minimum values are obtained by sorting the current solution. The numerical solution is performed on the resulting non-uniform grid in x. One problem with this equidistribution strategy is that it may lead to non-smooth mesh configurations. This is remedied by bounding the ratio of adjacent mesh spacings as follows

$$\frac{1}{A} \leq \frac{h_i}{h_{i-1}} \leq A \quad i = 1, 2, \ldots, N \tag{4.26}$$

where $A \geq 1$.

The mesh is refined by requiring that (4.24), (4.25) and (4.26) are satisfied. In intervals where these equations are not satisfied, a new mesh point is added. As in the method of Chong, interpolation is used to generate the solution on the new mesh. Smooke however, uses linear interpolation and notices a degeneration in the spatial truncation error. A novel approach used here however, is that the number of mesh points may be kept constant during a refinement. In such cases the new grid is obtained by extrapolating the existing grid to the next time step. This has an advantage if implicit time stepping is used since a Jacobian reevaluation may not be necessary after such a step if the number of mesh points remains the same. A common problem with grid extrapolation is that the mesh may become entangled. In this case Smooke simply reorders the new grid points consecutively.
In the approach used by Davis and Flaherty [14], the mesh equation and PDE are not solved simultaneously. They feel that the extra complexity involved in introducing a grid equation is not justified. They use a Galerkin Finite Element method with trapezoidal space-time elements. Finite Elements are used because in general they are more accurate than Finite Differences in the case of non-uniform grids.

The adaptive mesh algorithm attempts to place points in an optimal location and keep their positions optimal during the integration. The criterion for choosing the grid is the minimisation of the L_2 norm of the truncation error of the Finite Element method. This corresponds to equidistributing

$$h_i^{l+1} \mid U_{\epsilon_i}^{l+1} \mid \qquad x_i \leq \epsilon_i \leq x_{i+1} \tag{4.27}$$

where l is the degree of the particular polynomial trial functions being used. Using Finite Difference estimates of the higher solution derivatives the mesh is required to satisfy several differential relations. The associated equations are solved iteratively since they are nonlinear. The resulting grid may be extrapolated to future time steps and Davis and Flaherty find that zero-order extrapolation works well.

The main advantage of their approach is that the common problems experienced with the Moving Finite Element method (to be discussed in the next section) are alleviated, namely

- Computation may proceed on a non-optimal mesh.
- The grid distribution produced by the above equations does not suffer from mesh entanglement.

4.3.2 Mesh moving methods

In these methods a fixed number of mesh points are redistributed in order to resolve and follow evolving gradients in the solution. The distinguishing feature of quasi-Lagrangian methods is the treatment of the mesh locations as extra unknowns in the problem.

Consider the semidiscretisation of a parabolic/hyperbolic PDE resulting in the following system of ODEs.

$$\frac{dU_i}{dt} = f_i(t_i, U_1, U_2, ...U_{N+1}) \quad i = 1, \ldots, N+1 \tag{4.28}$$

Regarding the mesh points x_i as time dependent quantities means that temporal changes occur due to the equations (4.28) and also due to the movement of the grid. When the latter is taken into account the governing equations become

$$\frac{dU_i}{dt} = f + U_x x_t \tag{4.29}$$

where the subscripts have been dropped.

The second term above involves the ***grid speed*** x_t and this is provided by solving an appropriate equation modelling the grid motion. The use of an equation for the grid, however, further augments the original PDE system by one.

The various methods of mesh moving algorithms are distinguished primarily in the way they derive the mesh equations. Dorfi and Drury [16] describe a moving mesh method which alters the grid distribution so that the solution arc length is equidistributed over the spatial interval. Their grid equation is

$$\eta \propto \mathcal{R} \tag{4.30}$$

where $\mathcal{R}$ is some measure of the resolution of the solution and η is the grid point concentration. The solution arc length is chosen as the measure of resolution giving

$$\eta \propto \sqrt{1 + (\frac{dy}{dx})^2}$$

To avoid excessive grid distortion the condition

$$\frac{\alpha}{1+\alpha} \leq \frac{h_i}{h_{i-1}} \leq \frac{1+\alpha}{\alpha}$$

is imposed where α is some constant which can be described as the grid ***rigidity***. This condition is incorporated into the grid equation by spatially smoothing the right hand side of (4.30) as follows

$$\eta_i \propto \sum_j \mathcal{R} \left(\frac{\alpha}{1+\alpha} \right)^{i-j}$$

This results eventually in the grid equation

$$\frac{\eta_{i-1}}{\mathcal{R}_{i-1}} = \frac{\eta_i}{\mathcal{R}_i}$$

Coyle et al. [11] analyse several mesh equidistribution strategies for time-dependent PDEs. The approach used here is to construct a mesh moving equation which, when discretised, yields a system of ODEs for the mesh velocities. The mesh equation and PDE(s) may be solved simultaneously at each time step or mesh configurations, generated at a previous time level, may be extrapolated to a later time. Madsen [32] derives a variable mesh strategy suitable for explicit time integration schemes. The key point here is that a certain minimum mesh spacing must be maintained since the stability in time depends on this quantity. Madsen develops a grid equation based on a "mesh function," m.

$$\frac{d\dot{h}_i}{dt} = \frac{dx_{i+1}}{dt} - \frac{dx_i}{dt} = M - m^j$$

where h_i is the $i^{th.}$ mesh spacing and $M = \sum_{i=1}^{N} m^j / N$ is the average value of the mesh function over the entire mesh.

Various choices of mesh function are possible and Madsen describes a sub-equidistributing mesh strategy where the mesh function is chosen to be a weighted average of five different functions.

$$
\begin{aligned}
m_1^i &= \mid f_h - f_{2h} \mid_i / T_1 \\
m_2^i &= \mid U_{i+1} - U_i \mid / T_2 \\
m_3^i &= \mid (U_x)_{i+1} - (U_x)_i \mid / T_3 \\
m_4^i &= \mid K_{i+1} - K_i \mid / T_4 \\
m_5^i &= \mid x_{i+1} - x_i \mid / T_5
\end{aligned}
$$

Here, f_h and f_{2h} are estimates of the PDE right hand side for the present mesh, and for one which is twice as coarse. The K_i are estimates of the solution curvature and T_1, T_2, etc. normalise the individual mesh function values.
Rather than implement a full equidistribution strategy, Madsen opts for sub-equidistributing the mesh function on the grounds that much the same effectiveness is achieved with less expense. He presents an effective method of controlling the minimum mesh separation by altering the value of the mesh function at those mesh points which are in danger of becoming critically close so that they no longer have a tendency to move. The excess value of the mesh function thus removed is then redistributed over the other mesh points in proportion to their separation in excess of the minimum. He considers the overall algorithm to be robust and non-stiff in nature.

Moving Finite Elements

These methods are actually the same as the mesh moving methods of the last section. However, the notion of coupling the grid with the equation being solved is developed even further.

Rather than solving the problem with standard methods of discretisation and augmenting the PDE system with the grid equation, the Moving Finite Element (MFE) method utilises a method of solution which treats the mesh locations, and also the solution at these locations, as the unknowns to be determined. For this reason the MFE method does not share the properties of other moving grid methods and thus a separate description is justified.

The method was introduced by Miller and Miller [36] and further improved by Miller [35]. It was then extended to one-dimensional PDE systems by Gelinas, Dos and Miller [23]. For the scalar evolutionary PDE

$$U_t - L(U) = 0$$

the standard approach is to use the semidiscrete Galerkin or other Finite Element method to produce the approximate solution

$$v(x,t) = \sum_{i=1}^{N+1} a_i(t)\alpha_i(x) \tag{4.31}$$

The a_i represent the amplitudes of the trial functions α_i at the fixed nodes of the grid. In the MFE method the nodal positions, denoted for example by s_i, are allowed to be time dependent in the same way as the nodal amplitudes, a_i. Thus (4.31) becomes

$$v(x,t) = \sum_{i=1}^{N+1} a_i(t)\alpha_i(x, s(t)) \tag{4.32}$$

The method is usually associated with piecewise linear trial functions since they are the most basic form of elements and are simple to implement. Determination of the 2(N+1) unknowns $(a_i, s_i, i = 1, 2, \ldots, N+1)$ is achieved by minimising the L_2 norm of the residual

$$\| v_t - L(v) \|_{L_2}$$

with respect to the time derivative of the parameters a_i, s_i. This gives two variational equations for the α and β.

$$\begin{aligned} < v_t - L(v), \alpha_i > &= 0 \\ < v_t - L(v), \beta_i > &= 0 \end{aligned}$$

This corresponds to 2(N+1) equations for the nodal unknowns and positions and they are called the Moving Finite Element equations. Writing these equations in ODE form gives

$$A(y)\dot{y} = g(y) \tag{4.33}$$

Note the difference between the form of (4.33) and the corresponding ODE system obtained by applying the classical method of lines. See section 3.1.1. Miller and Miller [36] noticed that the parameterisation became degenerate in some cases ie. the β_i became dependent on the α_i. To remove degeneracy they introduced internodal viscosities into the parameterisation. Miller [35] further introduced internodal spring functions which became infinitely stiff as the nodes tended towards a minimum permissible separation. This removed problems associated with grid entanglement which the method had been known to possess. The method generally has been criticised for its stiffness and its failure in circumstances where the mesh becomes non-optimal. The method of Davis and Flaherty [14], discussed in the last section, was motivated by such problems with the MFE method. For an outline of the various improvements possible over the standard approach see Miller [34].

The basic characteristics of MFE methods are

- The number of nodes is constant.
- Nodes are not connected with a particular solution property as in the other mesh moving methods.
- The new grid coordinates and the solution are calculated simultaneously.

Given the extra effort needed in order to regularise the parameterisation and prevent mesh entanglement, the Moving Finite Element method succeeds in placing the nodes effectively. Wathen and Baines [56] have analysed the structure of the Moving Finite Element matrix A in (4.33) and they detail efficient methods for calculating its inverse for various types of problems.

Chapter 5

Choice of Algorithms and Their Implementation

In this chapter three adaptive mesh algorithms are studied in detail with emphasis given to their implementation strategies. These algorithms shall be fully implemented in Chapter 6 (to follow) where their effectiveness in the solution of parabolic problems will be examined.

The choice of an effective adaptive mesh algorithm is somewhat complicated by the great diversity of approaches used in such methods. In the case of parabolic problems, however, adaptive spatial meshing strategies are likely to be used in conjunction with existing temporal integration methods. Thus it is natural to expect good adaptive meshing software to possess the acknowledged qualities of good ODE integrators.

For parabolic problems in particular, the following criteria are regarded as necessary standards for high quality software.

1. **Versatility**.
2. **Robustness**.
3. **Efficiency**.

The first criterion is important because of the existing variety of parabolic-type problems. A successful code will have to cope with many different problem situations. This requirement affects considerably the choice of suitable adaptive mesh algorithms. In the last chapter it was seen that many such algorithms require some form of "tuning" to the problem situation at hand. This is clearly in conflict with the need for general applicability. A likely choice of algorithm will therefore require a minimal amount of such problem-specific information.

The second criterion determines to a large degree how popular the code will be among users. No matter how advanced a particular algorithm may be, if it is not implemented robustly then it will not be widely accepted among the user community. In problematic situations, therefore, the code must avert the possibility of failure. Usually robustness can be improved by applying heuristic approaches. For instance, in the original BDF code of Gear [21,22], alteration of the step size and order were disabled for several steps following the last alteration, so that the possibility of an error buildup was avoided. In the adaptive mesh algorithms of Chapter 4 some were inherently robust whereas others required adjustment. Again the need for tuning usually conflicts with the requirement for robustness. However, tuning parameters do arise which only effect the efficiency of the implementation.

The third criterion, efficiency, is also of importance to the success of an algorithm. This is especially true in the case of adaptive mesh algorithms since the whole point of employing such methods is to improve the accuracy of solutions and reduce the computer time required to produce them. An adaptive mesh algorithm should be much more efficient than its non-adaptive counterparts. It is not always obvious, however, if this is in fact the case. For example, Chong's approach [8] has been criticised for being very expensive to implement. Also in some strategies based on transformation, for example the transformation suggested by Braddock and Noye [4] in Chapter 4 section 4.2.3, the transformed equation may be more costly to solve than the original problem.

Considering the many algorithms at our disposal described in Chapter 4, two such algorithms have been selected which represent effective approaches toward adaptive meshing. The selected algorithms are also quite different from each other and thus a comparison between two distinct adaptive mesh strategies is possible.

The suitability of each algorithm for incorporation into existing software for parabolic problems depends on how well criteria 1, 2 and 3 are met. In the following chapter, the relative merits and/or deficiencies of each algorithm, with respect to the above criteria, will be examined.

The chosen algorithms are as follows.

- **Method of Verwer et al.** [53] (based upon an approach used by Dorfi and Drury [16]).
  This is a conventional moving mesh approach where an equation governing the grid motion is coupled with the original partial differential equation. The "grid" equation is based on a spatial equidistribution principle. In addition, spatial and temporal smoothing is applied to the grid motion which results in better control over the nodes and favourable consequences for the temporal integration. The method can be easily incorporated into a standard "method of lines" approach.

- **The Moving Finite Element method of Miller** [36,35].
  This is a version of the moving mesh method in which the grid positions and solution are unknowns in the Finite Element formulation. The mesh movement is guided so that spatial resolution of the solution is maintained. Thus the nodes are not connected with a particular solution property as in the other moving mesh methods. The original method has been criticised for its stiffness and its failure in circumstances where the grid becomes non-optimal. However, the use of so-called "penalty" functions (Miller [34]) greatly enhances the performance of the method. As in the last method the Moving Finite Element method is easily incorporated into a "method of lines" approach.

In the following sections each algorithm is derived fully and their implementation details discussed in detail.

5.1 Method of Verwer et al.

This is a mesh moving approach where the original PDE is expressed in Lagrangian form and is further augmented by a "grid" equation.

For the PDE

$$\frac{\partial u}{\partial t} = f(u) \quad 0 \leq x \leq 1 \quad t \geq 0 \tag{5.1}$$

Consider a transformation from coordinates (x,t) to new coordinates (s,t). Temporal changes in u now occur due to the differential operator f(u) and the motion of the grid.

$$\begin{aligned} \frac{\partial u(s,t)}{\partial t} &= \frac{\partial u(x,t)}{\partial t} + \frac{\partial u(x,t)}{\partial x}\frac{\partial x(t)}{\partial t} \\ \frac{\partial u(s,t)}{\partial t} &= f(u) + \frac{\partial u(x,t)}{\partial x}\frac{\partial x(t)}{\partial t} \end{aligned} \tag{5.2}$$

The term $\partial x(t)/\partial t$ appearing in the Lagrangian form (5.2) is termed the grid speed. The grid and solution evolves continuously with the mesh movement being coupled to the solution via the grid speed term in (5.2). Furnishing this term is achieved by solving an appropriate equation for the grid motion. The original PDE (5.1) therefore becomes more complicated itself and must also be solved together with the equation for the grid.

Verwer et al. [53] use a grid equation derived by Dorfi and Drury [16], based on spatial equidistribution. The form of the equation is due to Dorfi and Drury [16]. Basically the equation consists of a proportionality between measures of achieved and desired resolution of the solution. As a measure of achieved resolution the point concentration suffices. For a general non-uniform grid

$$\Pi_N : 0 = x_0 < x_1 < x_2 < \ldots < x_N = 1$$

on the spatial interval $x \in [0,1]$ the point concentration is defined as

$$n_i = \frac{1}{x_{i+1} - x_i} \qquad 0 \leq i \leq N-1$$

The measure of resolution, R, is chosen to be the solution arc length as in White's approach. This is approximated by

$$R_i = \sqrt{1 + \left(\frac{u_{i+1} - u_i}{x_{i+1} - x_i}\right)^2} \quad 0 \le i \le N-1 \tag{5.3}$$

Thus the basic grid equation becomes

$$n_i \propto R_i \quad 0 \le i \le N-1 \tag{5.4}$$

Eliminating the proportionality constant gives

$$\frac{n_{i-1}}{R_{i-1}} = \frac{n_{i+1}}{R_{i+1}} \quad 1 \le i \le N-1 \tag{5.5}$$

As was noted in section 4.2.4 in Chapter 4, meshes generated using such an equidistribution equation require regularisation in order to be effective. This achieved in this case by applying both spatial and temporal smoothing to the basic grid equation (5.5).

5.1.1 Spatial smoothing technique

Spatial smoothing is achieved by restricting changes in the local grid spacing requiring

$$\frac{\alpha}{\alpha+1} \le \frac{n_{i-1}}{n_i} \le \frac{\alpha+1}{\alpha} \quad 1 \le i \le N-1$$

where α is a "grid rigidity" parameter. For instance, if $\alpha = 1$ then the relative increase/decrease in grid point concentration over two adjacent mesh intervals is at most a factor of 2. This restriction is similar to that derived in section 4.2.1 in order to preserve second order accuracy using standard Finite Differences on a non-uniform grid.

At points of the mesh successively distant from the point x_i the point concentrations are as follows

$$\left(\frac{\alpha}{\alpha+1}\right)^2 n_i, \left(\frac{\alpha}{\alpha+1}\right) n_i, n_i, \left(\frac{\alpha+1}{\alpha}\right) n_i, \left(\frac{\alpha+1}{\alpha}\right)^2 n_i$$

Thus in equation (5.4), n_i can be replaced by its spatially smoothed value giving

$$n_i \propto \sum_{j=0}^{N} \left(\frac{\alpha}{\alpha+1}\right)^{|j-i|} R_j \tag{5.6}$$

This smoothing "kernel" is the Green's function associated with the central difference operator

$$1 - \alpha(\alpha+1)\delta^2$$

This allows the left hand side of (5.6) to be replaced by central differences as follows.

$$\tilde{n}_i = n_i - \alpha(\alpha+1)(n_{i+1} - 2n_i + n_{i-1}) \propto R_i \quad 1 \le i \le N-2 \tag{5.7}$$

For the boundaries, "zero concentration gradient" conditions are chosen, namely

$$n_0 = n_1 \tag{5.8}$$
$$n_N = n_{N-1} \tag{5.9}$$

and for the first and last nodes, respectively, (5.7) becomes

$$\begin{aligned} \tilde{n}_0 &= n_0 - \alpha(\alpha+1)(n_1 - n_0) \propto R_i \\ \tilde{n}_{N-1} &= n_N - \alpha(\alpha+1)(n_{N-2} - n_{N-1}) \propto R_i \end{aligned}$$

Using these spatially smoothed values of the point concentrations means that the grid equation (5.5) becomes

$$\frac{\tilde{n}_{i-1}}{R_{i-1}} = \frac{\tilde{n}_i}{R_i} \quad 1 \le i \le N-1 \tag{5.10}$$

Since n_{i-1} and n_i appear simultaneously in (5.10) then its corresponding difference replacement will be five-point in space.

The matrix associated with the above smoothing of the point concentration is given by

$$\mathbf{Rn} = \tilde{\mathbf{n}} \tag{5.11}$$

where $\mathbf{n} = [n_0, n_1, \ldots, n_{N-1}]^T$ and $\tilde{\mathbf{n}} = [\tilde{n}_0, \tilde{n}_1, \ldots, \tilde{n}_{N-1}]^T$.

R is symmetric and positive definite and so the $\tilde{n}_i$ are uniquely determined by

$$\mathbf{n} = \mathbf{R}^{-1}\tilde{\mathbf{n}} \tag{5.12}$$

The spatial grid smoothing involves solving (5.12) for the point concentrations $\tilde{n}_i$. The smoothing procedure given by (5.7) is a second order nonhomogeneous linear difference equation. The auxiliary equation for the associated homogeneous problem is (letting $n_i = r^i$)

$$\begin{aligned} r^i - \alpha(\alpha+1)(r^{i+1} - 2r^i + r^{i-1}) &= 0 \\ r - \alpha(\alpha+1)(r^2 - 2r + 1) &= 0 \\ r^2(-\alpha(\alpha+1)) + r(1 + 2\alpha(\alpha+1)) - \alpha(\alpha+1) &= 0 \end{aligned}$$

Solving this quadratic equation for r gives

$$\begin{aligned} r &= \frac{-1 - 2\alpha(\alpha+1) \pm 2\alpha + 1}{-2\alpha(\alpha+1)} \\ r &= \frac{\alpha}{\alpha+1} \pm \frac{\alpha+1}{\alpha} \end{aligned}$$

By the principle of superposition for linear equations a general solution to the homogeneous problem is given by

$$n_{i_{hom}} = A_1\left(\frac{\alpha}{\alpha+1}\right)^i + A_2\left(\frac{\alpha+1}{\alpha}\right)^i \qquad 0 \le i \le N-1 \tag{5.13}$$

A particular solution of the homogeneous equation is

$$n_{i_{par}} = \frac{1}{1+2\alpha}\sum_{j=1}^{N-1}\left(\frac{\alpha}{\alpha+1}\right)^{|i-j|}\tilde{n}_j \qquad 0 \le i \le N$$

The general solution to (5.7) is therefore the sum of the general and particular solutions (5.13) and (5.1.1).

$$n_i = A_1\left(\frac{\alpha}{\alpha+1}\right)^i + A_2\left(\frac{\alpha+1}{\alpha}\right)^i \frac{1}{1+2\alpha}\sum_{j=1}^{N-1}\left(\frac{\alpha}{\alpha+1}\right)^{|i-j|}\tilde{n}_j \qquad 0 \le i \le N - (5.14)$$

The constants A_1 and A_2 are determined by the boundary conditions given in (5.8) and (5.9). Neglecting the boundaries (5.14) becomes

$$n_i = \frac{1}{1+2\alpha} \sum_{j=1}^{N-1} \left(\frac{\alpha}{\alpha+1}\right)^{|i-j|} \tilde{n}_j \quad 0 \le i \le N-1$$

The expression

$$\left(\frac{\alpha}{\alpha+1}\right)^{|i-j|}$$

is the "smoothing kernel" and

$$0 \le \frac{\alpha}{\alpha+1} \le 1$$

Thus n_i always remains positive. This means that the spatial grid smoothing process cannot lead to node crossing and that the original grid condition (5.1.1) is indeed desirable.

5.1.2 Temporal smoothing technique

Temporal smoothing is introduced by replacing the algebraic system (5.10) by a system of differential equations

$$\left(\tilde{n}_{i-1} + \tau\frac{\partial \tilde{n}_{i-1}}{\partial t}\right)/R_{i-1} = \left(\tilde{n}_i + \tau\frac{\partial \tilde{n}_i}{\partial t}\right)/R_i \quad 1 \le i \le N-1 \qquad (5.15)$$

The presence of the time derivatives of the point concentrations means that the grid will not adjust immediately to the values of R but will do so over a time interval τ. Thus τ is a parameter which causes a time delay in the equidistribution equation resulting in a smoother evolution of the grid in time.

Equation (5.15) is derived from the corresponding relation

$$\tilde{n}_i(t) + \tau\frac{d\tilde{n}_i(t)}{dt} = c(t)R_i(t) \qquad (5.16)$$

This a linear first order nonhomogeneous ordinary differential equation which can be solved for $\tilde{n}_i(t)$ as follows. From (5.16)

$$\begin{aligned} \frac{d\tilde{n}_i(t)}{dt} + \frac{\tilde{n}_i(t)}{\tau} &= \frac{c(t)R_i(t)}{\tau} \qquad (5.17) \\ \left(\frac{\tilde{n}_i(t) - c(t)R_i(t)}{\tau}\right) dt + d\tilde{n}_i(t) &= 0 \end{aligned}$$

Assuming an integrating factor F(t) exists for this equation then

$$F(t)\left(\frac{\tilde{n}_i(t) - c(t)R_i(t)}{\tau}\right) dt + F(t)d\tilde{n}_i(t) = 0$$

Equation (5.18) is an exact differential equation [1] if

$$\frac{\partial}{\partial \tilde{n}_i}\{F(t)\left(\frac{\tilde{n}_i(t) - c(t)R_i(t)}{\tau}\right)\} = \frac{dF(t)}{dt}$$

This simplifies to

$$F(t)\frac{1}{\tau} = \frac{dF(t)}{dt}$$

By separation of variables we get

$$\begin{aligned} \frac{1}{\tau}\int dt &= \int \frac{1}{F(t)} dF \\ \frac{t}{\tau} &= ln \mid F(t) \mid \end{aligned}$$

Thus F(t) = exp(t/τ) is the required integrating factor. Multiplying (5.17) by this integrating factor gives

[1]The expression P(x,u)du + Q(x,u)du is an exact differential, ie. the differential of a function F(x,u), if

$$\frac{\partial P}{\partial u} = \frac{\partial Q}{\partial x}$$

$$\exp\frac{t}{\tau}\left(\frac{d\tilde{n}_i(t)}{dt}+\frac{\tilde{n}_i(t)}{\tau}\right) = \exp\frac{t}{\tau}\frac{c(t)R_i(t)}{\tau}$$

which simplifies to

$$\frac{d}{dt}\{\tilde{n}_i(t)\exp\frac{t}{\tau}\} = \frac{1}{\tau}\exp\frac{t}{\tau}c(t)R_i(t)$$

Integrating both sides gives

$$\begin{aligned}\tilde{n}_i(t)\exp(\frac{t}{\tau}) &= \frac{1}{\tau}\int_0^t \exp(\frac{t}{\tau})c(s)R_i(s)ds + C \\ \tilde{n}_i(t) &= \exp(\frac{t}{\tau})\left[C+\frac{1}{\tau}\int_0^t \exp^{(\frac{t}{\tau})}c(s)R_i(s)ds\right] \qquad (5.18)\end{aligned}$$

At t=0 this reduces to $\tilde{n}_i(0) = C$ and thus (5.18) becomes

$$\tilde{n}_i(t) = \exp(\frac{t}{\tau})\left[\tilde{n}_i(0)+\frac{1}{\tau}\int_0^t \exp(\frac{t}{\tau})c(s)R_i(s)ds\right] \qquad (5.19)$$

where the initial smoothed point densities, $\tilde{n}_i$, are determined from (5.7).

Let us now examine (5.19) at a time t following a time step of Δt.

$$\tilde{n}_i(t) = \exp^{(\frac{\Delta t}{\tau})}\left[\tilde{n}_i(t-\Delta t)+\frac{1}{\tau}\int_{t-\Delta t}^t \exp^{(\frac{s-t}{\tau})}c(s)R_i(s)ds\right] \quad (5.20)$$

for

$$t \geq \Delta t \qquad 0 \leq i \leq N$$

The value of $\tilde{n}_i(t)$ is the sum of $\exp(\frac{\Delta t}{\tau})$ and the weighted average of $c(s)R_i(s)$ over the interval $[t-\Delta t, t]$. The weighting depends on the size of τ. As $\tau \rightarrow \infty$ then $\tilde{n}_i(t) \rightarrow \tilde{n}_i(t-\Delta t)$. Thus for large τ values the grid remains stationary. As $\tau \rightarrow 0$ then $\tilde{n}_i(t) \rightarrow c(t)R_i(t)$ and the point concentration adjusts solely to the present value of $c(t)R_i(t)$ and no temporal smoothing occurs.

5.1.3 Implementation

Taking the grid equation (5.15)

$$\left(\tilde{n}_{i-1} + \tau\frac{\partial \tilde{n}_{i-1}}{\partial t}\right)/R_{i-1} = \left(\tilde{n}_i + \tau\frac{\partial \tilde{n}_i}{\partial t}\right)/R_i \qquad 1 \le i \le N-1$$

and substituting

$$\mu = \alpha(\alpha+1)$$

$$\begin{aligned}
\tilde{n}_0 &= -\mu n_1 + (1+\mu)n_0 \qquad i = 1\\
\tilde{n}_i &= -\mu n_{i+1} + (1+2\mu)n_i - \mu n_{i-1} \qquad 2 \le i \le N-2\\
\tilde{n}_{N-1} &= -\mu n_{N-2} + (1+\mu)n_{N-1} \qquad i = N-1
\end{aligned}$$

gives

$$[-\mu n_1 + (1+\mu)n_0 + \tau(-\mu n_1' + (1+\mu)n_0')]/R_0 +$$
$$[-\mu n_2 + (1+2\mu)n_1 - \mu n_0 + \tau(-\mu n_2' + (1+2\mu)n_1' - \mu)]/R_1 = 0$$

$$[-\mu n_i + (1+2\mu)n_{i-1} - \mu n_{i-2} + \tau(-\mu n_i' + (1+2\mu)n_{i-1}' - nn_{i-2}')]/R_{i-1} -$$
$$[-\mu n_{i+1} + (1+2\mu)n_i - \mu n_{i-1} +$$
$$\tau(-\mu n_{i+1}' + (1+2\mu)n_i' - \mu N_{i-1}')]/R_i = 0 \qquad 2 \le i \le N-2$$

$$[-\mu n_{N-1} + (1+2\mu)n_{N-2} - \mu n_{N-3} + \tau(-\mu n_{N-1}' + (1+2\mu)n_{N-2}' - \mu n_{N-3}')]/R_{N-2} -$$
$$[-\mu n_{N-2} + (1+\mu)n_{N-1} + \tau(-\mu n_{N-2}' + (1+\mu)n_{N-1}')]/R_{N-1}$$

Next substituting

$$n_i = \frac{1}{\Delta x_i}; \qquad n_i' = -\frac{\Delta x_i'}{(\Delta x_i)^2}; \qquad \Delta x_i' = x_{i+1}' - x_i'$$

gives (after some simplification)

$$\begin{aligned}
&\tau\left[\frac{1+\mu}{R_0(\Delta X_0)^2}+\frac{\mu}{R_1(\Delta X_0)^2}\right]X_0' - \\
&\tau\left[\frac{\mu}{R_0(\Delta X_1)^2}+\frac{1+\mu}{R_0(\Delta X_0)^2}+\frac{1+2\mu}{R_1(\Delta X_1)^2}+\frac{\mu}{R_1(\Delta X_0)^2}\right]X_1' + \\
&\tau\left[\frac{\mu}{R_1(\Delta X_2)^2}+\frac{1+2\mu}{R_1(\Delta X_1)^2}+\frac{\mu}{R_0(\Delta X_1)^2}\right]X_2' - \\
&\tau\left[\frac{\mu}{R_1(\Delta X_2)^2}\right]X_3' \\
&=\left[\frac{\mu}{\Delta X_1}-\frac{1+\mu}{\Delta X_0}\right]/R_0+\left[-\frac{\mu}{\Delta X_2}+\frac{1+2\mu}{\Delta X_1}-\frac{\mu}{\Delta X_0}\right]/R_1
\end{aligned}$$

$$\begin{aligned}
- \quad &\tau\left[\frac{\mu}{R_{i-1}(\Delta X_{i-2})^2}\right]X_{i-2}' + \\
&\tau\left[\frac{\mu}{R_i(\Delta X_{i-1})^2}+\frac{1+2\mu}{R_{i-1}(\Delta X_{i-1})^2}+\frac{\mu}{R_{i-1}(\Delta X_{i-2})^2}\right]X_{i-1}' - \\
&\tau\left[\frac{\mu}{R_i(\Delta X_{i-1})^2}+\frac{1+2\mu}{R_i(\Delta X_i)^2}+\frac{1+2\mu}{R_{i-1}(\Delta X_{i-1})^2}+\frac{\mu}{R_{i-1}(\Delta X_i)^2}\right]X_i' + \\
&\tau\left[\frac{\mu}{R_i(\Delta X_{i+1})^2}+\frac{1+2\mu}{R_i(\Delta X_i)^2}+\frac{\mu}{R_{i-1}(\Delta X_i)^2}\right]X_{i+1}' - \\
&\tau\left[\frac{\mu}{R_i(\Delta X_{i+1})^2}\right]X_{i+2}' \\
&=\left[-\frac{\mu}{\Delta X_{i+1}}+\frac{1+2\mu}{\Delta X_i}-\frac{\mu}{\Delta X_{i-1}}\right]/R_i-\left[-\frac{\mu}{\Delta X_i}+\frac{1+2\mu}{\Delta X_{i-1}}-\frac{\mu}{\Delta X_{-2i}}\right]/R_{i-1}
\end{aligned}$$

$$\begin{aligned}
- \quad &\tau\left[\frac{\mu}{R_{N-3}(\Delta X_{N-3})^2}\right]X_{N-3}' + \\
&\tau\left[\frac{\mu}{R_{N-1}(\Delta X_{N-2})^2}+\frac{\mu}{R_{N-2}(\Delta X_{N-3})^2}+\frac{1+2\mu}{R_{N-2}(\Delta X_{N-2})^2}\right]/X_{N-1} - \\
&\tau\left[\frac{\mu}{R_{N-2}(\Delta X_{N-1})^2}+\frac{\mu}{R_{N-1}(\Delta X_{N-2})^2}+\frac{\mu}{R_{N-1}(\Delta X_{N-1})^2}+\frac{1+2\mu}{R_{N-2}(\Delta X_{N-2})^2}\right]X_{N-3}' - \\
&\tau\left[\frac{\mu}{R_{N-2}(\Delta X_{N-1})^2}+\frac{1+\mu}{R_{N-1}(\Delta X_{N-1})^2}\right]X_N' \\
&=\left[\frac{1+\mu}{\Delta X_{N-1}}-\frac{\mu}{\Delta X_{N-2}}\right]/X_{N-1}+\left[\frac{\mu}{\Delta X_{N-1}}-\frac{1+2\mu}{\Delta X_{N-2}}+\frac{\mu}{\Delta X_{N-3}}\right]/R_{N-2}
\end{aligned}$$

Note that the above system is five-point coupled in space (excluding the first and last equations). The system may be written in implicit ODE form

$$\tau B(X,U)X' = G(X,U)$$

The matrix B is pentadiagonal and of dimension (N-1)(N-1). In the case of zero temporal smoothing the system (5.1.3) reduces to the algebraic system

$$G(X,U) = 0$$

The original problem differential is coupled with the grid equation in the following way. Taking the Lagrangian form of the original PDE given by (5.2)

$$U' - X'U_x = F(X,U)$$

and combining this with (5.1.3) gives the following system of 2(N-1) equations.

$$\begin{aligned} \tau B(X,U)X' &= G(X,U) \\ U' - X'U_x &= F(X,U) \end{aligned} \qquad (5.21)$$

The terms U_x and F above are approximated by centered Finite Differences based on a non-uniform grid as derived in Chapter 3, subsection 3.1.1. Reordering the above system so that the unknowns appear as,

$$.........U_{i-1}, X_{i-1}, U_i, X_i, U_{i+1}, X_{i+1}..........$$

means that the left hand side matrix in (5.21) has a banded form with band-width 9. The form of the system (5.21) for N=5 mesh spacings (presuming a single parabolic equation with Dirichlet boundary data) is shown in Figure 5.1. Incorporating the boundary values of U and X means that the system (5.21) is of order 2(N+1). The equations corresponding to X_0' and X_N' are trivial since these points are fixed at all times. However, the equations corresponding to U_0' and U_N' depend on the boundary conditions of the PDE.

Solving the system (5.21) requires an ODE solver capable of handling linearly implicit differential and/or algebraic equations. In the following chapter the above method is implemented using the LSODI [25] based code of the NAG [39] routine D02NHF.

$$\begin{pmatrix}
* & . & . & . & . & & & & & & & \\
. & * & . & . & . & . & & & & & & \\
. & . & * & * & . & . & . & & & & & \\
. & * & . & * & . & * & . & * & & & & \\
. & . & . & . & * & * & . & . & . & & & \\
 & * & . & * & . & * & . & * & . & * & & \\
 & & . & . & . & . & * & * & . & . & . & \\
 & & & * & . & * & . & * & . & * & . & * \\
 & & & . & . & . & . & . & * & * & . & . \\
 & & & & & * & . & * & . & * & . & * \\
 & & & & & & . & . & . & . & * & . \\
 & & & & & & & . & . & . & . & *
\end{pmatrix}
\begin{pmatrix} \dot{U}_0 \\ X_0 \\ U_1 \\ X_1 \\ U_2 \\ X_2 \\ U_3 \\ X_3 \\ U_4 \\ X_4 \\ U_5 \\ X_5 \end{pmatrix}
=
\begin{pmatrix} * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \end{pmatrix}
\tag{5.22}$$

Figure 5.1: Structure of the ODE system for Verwer's method in the case of a single parabolic equation (with Dirichlet boundary data).

5.2 Moving Finite Elements.

In Chapter 3, section 3.1.2, the method of weighted residuals was outlined. For an equation of the form

$$\frac{\partial u}{\partial t} = L(u) \tag{5.23}$$

this leads to an approximate solution (or trial solution) of the form

$$u(x,t) = \sum_{j=0}^{N} a_j(t)\alpha_j(x) \tag{5.24}$$

Here, the α_j form a linearly independent set of known analytic functions, called trial functions, and the a_j represent the amplitudes of the approximate solution, y(x,t), at the nodes of the spatial mesh.

$$\Pi_N : 0 = x_0 < x_1 < x_2 < \ldots < x_N = 1$$

The Moving Finite Element (MFE) method adopts the same approach as above but the nodal positions are allowed to be time dependent. Equation (5.24) thus becomes

$$u(x,t) = \sum_{j=0}^{N} a_j(t)\alpha_j(x,s(t)) \qquad (5.25)$$

where s is a vector of time-dependent nodal positions. Partial differentiation with respect to time gives

$$\begin{aligned} \frac{\partial u}{\partial t} &= \sum_{j=0}^{j=N}[\frac{\partial u}{\partial a_j}\frac{\partial a_j}{\partial t} + \frac{\partial u}{\partial s_j}\frac{\partial s_j}{\partial t}] \\ \frac{\partial u}{\partial t} &= \sum_{j=0}^{j=N}[\alpha_j(x,s(t))\frac{\partial a_j}{\partial t} + \beta_j(x,a(t),s(t))\frac{\partial s_j}{\partial t}] \qquad (5.26) \end{aligned}$$

where $\beta(x,a(t),s(t)) = \frac{\partial u}{\partial s_j}$.

Notice that the form of (5.26) is similar to the Lagrangian form of (5.23) given by (5.2) which indicates a possible Lagrangian interpretation of the Moving Finite Element Method.

The trial functions α_i and the trial solution u belong to the same space of functions and the simplest and easiest form for use in the Moving Finite Element method is the piecewise linear (hat) function and solution shown in Figure 5.2.

In Figure 5.2 the piecewise linear (hat) trial functions are given by

$$\alpha_j(x) = \begin{cases} 0 & \text{if } x \leq s_{j-1} \\ 1 & \text{if } x = s_j \\ \frac{x-s_{j-1}}{s_j-s_{j-1}} & \text{if } s_{j-1} \leq x \leq s_j \\ \frac{s_{j+1}-x}{s_{j+1}-s_j} & \text{if } s_j \leq x \leq s_{j+1} \\ 0 & \text{if } x \geq s_{j+1} \end{cases} \qquad (5.27)$$

An equation for the piecewise linear trial solution u on the interval $[s_{j-1}, s_j]$ is as follows.

$$u - a_{j-1} = m_j(x - s_{j-1})$$

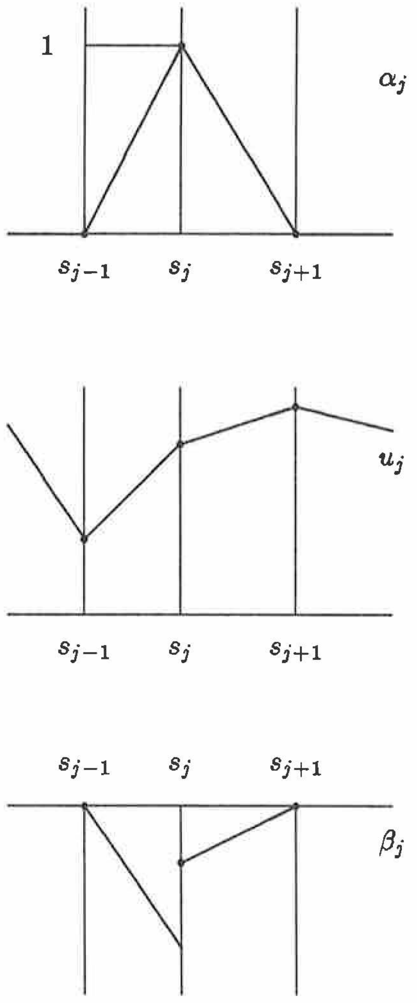


Figure 5.2: The functions α_j, u_j and β_j for the Moving Finite Element method.

$$\begin{aligned} u &= a_{j-1} + m_j(x - s_{j-1}) \\ u &= \alpha_{j-1} + \left(\frac{a_j - a_{j-1}}{s_j - s_{j-1}}\right)(x - s_{j-1}) \end{aligned}$$

Therefore the auxiliary trial function β_j can be expressed as follows.

$$\begin{aligned} \beta_j = \frac{\partial u}{\partial s_j} &= -\left(\frac{a_j - a_{j+1}}{s_j - s_{j-1}}\right)\left(\frac{x - s_{j-1}}{s_j - s_{j-1}}\right) \\ &= -m_j \alpha_j \end{aligned}$$

Likewise on the interval $x \in [s_j, s_{j+1}]$

$$\begin{aligned} u - a_j &= m_{j+1}(x - s_j) \\ u &= a_j + m_{j+1}(x - s_j) \\ u &= \alpha_j + \left(\frac{a_{j+1} - a_j}{s_{j+1} - s_j}\right)(x - s_j) \end{aligned}$$

$$\begin{aligned} \beta_{j+1} = \frac{\partial u}{\partial s_{j+1}} &= -\left(\frac{a_{j+1} - a_j}{s_{j+1} - s_j}\right)\left(\frac{x - s_j}{s_{j+1} - s_j}\right) \\ &= -m_j \alpha_j \end{aligned}$$

Thus the form of the function $\beta_j(x)$ is

$$\beta_j(x) = \begin{cases} 0 & \text{if } x \le s_{j-1} \\ -m_j \alpha_j(x) & \text{if } s_{j-1} \le x \le s_j \\ -m_j & \text{if } x = s_j \\ m_{j+1}\alpha_j(x) & \text{if } s_j \le x \le s_{j+1} \\ 0 & \text{if } x \ge s_{j+1} \end{cases} \tag{5.28}$$

which is shown in Figure 5.2.

The β_j may be regarded as secondary trial functions similar to the α_j and with the same local support.

The coefficients of the approximate solution (5.24) are determined by the method of weighted residuals (see Chapter 6). Using the Galerkin approach the L_2 norm of the residual

$$\left\| \frac{\partial u}{\partial t} - L(u) \right\|_{L_2}$$

with respect to the time derivatives of the parameters a_j, s_j is minimised. That is, the inner products of the weighted residuals are simultaneously set to zero.

$$\begin{aligned} \langle \frac{\partial u}{\partial t} - L(u), \alpha_j \rangle &= 0 \\ \langle \frac{\partial u}{\partial t} - L(u), \beta_j \rangle &= 0 \end{aligned}$$

Substituting $\frac{\partial u}{\partial t}$ from (5.26) gives

$$\begin{aligned} \langle \sum_{j=0}^{N} [\alpha_j \frac{da_j}{dt} + \beta_j \frac{ds_j}{dt}] - L(u), \alpha_j \rangle &= 0 \\ \langle \sum_{j=0}^{N} [\alpha_j \frac{da_j}{dt} + \beta_j \frac{ds_j}{dt}] - L(u), \beta_j \rangle &= 0 \end{aligned}$$

and finally

$$\begin{aligned} \sum_{j=0}^{N} \langle \alpha_j, \alpha_j \rangle \frac{da_j}{dt} + \sum_{j=0}^{N} \langle \alpha_j, \beta_j \rangle \frac{ds_j}{dt} - \langle \alpha_j, L(u) \rangle &= 0 \\ \sum_{j=0}^{N} \langle \beta_j, \alpha_j \rangle \frac{da_j}{dt} + \sum_{j=0}^{N} \langle \beta_j, \beta_j \rangle \frac{ds_j}{dt} - \langle \beta_j, L(u) \rangle &= 0 \end{aligned}$$

This corresponds to 2(N+1) equations for the nodal unknowns and positions and they are called the Moving Finite Element equations. Writing these equations in ODE form gives

$$A(u)\dot{u} = g(u) \tag{5.29}$$

where A has the block-tridiagonal form

$$\begin{pmatrix} A_1 & B_1 & & & & & \\ C_2 & A_2 & B_2 & & & & \\ & C_3 & A_3 & B_3 & & & \\ & & .. & .. & .. & & \\ & & & .. & .. & .. & \\ & & & & C_N & A_N & B_N \\ & & & & & C_{N+1} & B_{N+1} & A_{N+1} \end{pmatrix}$$

and u is an (N+1)2 vector of the form

$$\begin{pmatrix} a_1 \\ s_1 \\ a_2 \\ s_2 \\ . \\ . \\ . \\ . \\ a_{N+1} \\ s_{N+1} \end{pmatrix}$$

A(y) is a symmetric square matrix composed of N+1 (2x2) blocks of the form

$$A_i = \begin{pmatrix} <\alpha_i, \alpha_j> & <\alpha_i, \beta_j> \\ <\beta_i, \alpha_j> & <\beta_i, \beta_j> \end{pmatrix} \tag{5.30}$$

$$B_i = \begin{pmatrix} <\alpha_i, \alpha_j> & <\alpha_i, \beta_j> \\ <\beta_i, \alpha_j> & <\beta_i, \beta_j> \end{pmatrix} \tag{5.31}$$

$$C_i = \begin{pmatrix} <\alpha_i, \alpha_j> & <\alpha_i, \beta_j> \\ <\beta_i, \alpha_j> & <\beta_i, \beta_j> \end{pmatrix} \tag{5.32}$$

The vector g is composed of inner products of the spatial operator L(u) as follows

$$g(u) = (<\alpha_1, L(u)>, <\beta_1, L(u)>, ... <\alpha_{N+1}, L(u)>, <\beta_{N+1}, L(u)>)$$

5.2.1 Implementation

In [23] the frequently used inner products are derived fully. Expanding equations (5.2) gives the following.

$$\begin{aligned}
U'_{i-1} \quad & \left[\frac{\Delta X_i}{6}\right] + X'_{i-1}\left[-\frac{\Delta U_i}{6}\right] + \\
U'_i \quad & \left[\frac{\Delta X_i + \Delta X_{i+1}}{3}\right] + \\
X'_i \quad & \left[-\frac{\Delta U_i + \Delta U_{i+1}}{3}\right] + \\
U'_{i+1} \quad & \left[\frac{\Delta X_{i+1}}{6}\right] + X'_{i+1}\left[-\frac{\Delta U_{i+1}}{6}\right] =< \alpha_j, L(u) >
\end{aligned} \qquad (5.33)$$

$$\begin{aligned}
U'_{i-1} \quad & \left[-\frac{\Delta U_i}{6}\right] + X'_{i-1}\left[-\frac{m_i \Delta U_i}{6}\right] + \\
U'_i \quad & \left[-\frac{\Delta U_i + \Delta U_{i+1}}{3}\right] + \\
X'_i \quad & \left[\frac{m_i \Delta U_i + m_{i+1}\Delta U_{i+1}}{3}\right] + \\
U'_{i+1} \quad & \left[-\frac{\Delta U_{i+1}}{6}\right] + X'_{i+1}\left[\frac{m_{i+1}\Delta U_{i+1}}{6}\right] =< \beta_j, L(u) >
\end{aligned} \qquad (5.34)$$

This system is a linearly implicit ODE system and requires the use of a suitable ODE package. In the following chapter results for this adaptive mesh algorithm obtained using the NAG routine D02NHF will be presented.

The system (5.29) as it stands does not guarantee a reasonable adjustment of the spatial grid unlike the method of the previous section. This is for two reasons. Firstly, the mass matrix A may become numerically singular whenever the slopes of two mutually adjacent elements become equal (ie. collinearity). When this happens $m_i = m_{i+1}$ and by observing (5.28) one can see that the β_j are now proportional to the α_j. This however means that the corresponding diagonal block of A, A_i, will be zero making A singular. This phenomenon is termed "parallelism". The second difficulty encountered with the system (5.29) is the problem of node tangling which can happen when nodes drift too close together.

Miller [34] suggests the use of "penalty" functions in order to regularise the node motion. Instead of minimising

$$\frac{\partial u}{\partial t} - L(u)$$

we minimise

$$\frac{\partial u}{\partial t} - L(u) + \sum_{j=0}^{j=N} \left(\epsilon_j \Delta X_j' - S_j\right)^2$$

The ϵ-terms correspond to an internodal viscosity which penalises relative grid motion. This has the effect of rendering the mass matrix A diagonally dominant and thus invertible. The S-terms represent internodal spring functions which penalise the nodes from coming too close together.

The form of these penalty functions are, following Miller [34],

$$\epsilon_j^2 = \frac{C_1^2}{(\Delta x_j - d)} \qquad \epsilon_j S_j = \frac{C_2^2}{(\Delta X_j - d)^2}$$

The constants C_1, C_2 and d are user controlled parameters. The effect of these penalty functions is to add extra terms to the left and right hand sides of the system (5.29). The additional terms are

$$\epsilon_j^2 \Delta X_j' - \epsilon_{j+1}^2 \Delta X_{j+1}'$$

and

$$\epsilon_j S_j - \epsilon_{j+1} S_{j+1}$$

which are added to the left and right hand sides of the β equation (5.34), respectively. In the following chapter the merits and/or problems associated with these parameters will be examined.

Chapter 6

Numerical Investigations

In Chapter 5 the implementation details of two effective adaptive mesh strategies were described. Both strategies shall now be applied to the solution of four example parabolic problems. These examples are a representative sample of the wide variety of problems occurring in parabolic partial differential equations.

6.1 Algorithms summary

The main characteristics of the chosen adaptive mesh algorithms are given in Tables 6.2 and 6.3 below. For comparison purposes the characteristics of the conventional fixed grid MOL approach are also given in Table 6.1. In these descriptions it is assumed that each algorithm is applied to the solution of a single parabolic equation, discretised on the spatial interval using NN internal [1] grid points.

[1]by internal is meant "excluding the boundaries"

Method I. The Standard Method of Lines	
Type	Conventional fixed grid approach
Spatial discretisation	Finite Difference (with second order, centered differencing)
Temporal integrator	Explicit ODE system solver (NAG [39] routine D02NCF)
Jacobian structure	Tridiagonal matrix of order (NN+2)

Table 6.1: Characteristics of the Standard Method of Lines.

Method II. The Method of Verwer et al. [53,16]	
Type	Adaptive mesh approach from the class of "mesh moving" methods.
Implementation	Method of Lines
Spatial discretisation	Finite Difference (with second order centered differencing, based on a non-uniform grid)
Adaptive grid control	Based on spatial equidistribution of an arc length monitor function.
Temporal integrator	Linearly implicit ODE/DAE system solver (NAG [39] routine D02NHF)
Jacobian structure	Band matrix of order 2(NN+2) and band width 9
Method parameters	$\alpha(=\mathbf{1})$ monitor function regularisation parameter $\kappa(=\mathbf{2})$ spatial grid smoothing parameter τ temporal grid smoothing parameter

Table 6.2: Characteristics of the method of Verwer et al.

Method III. Moving Finite Elements [36,35]	
Type	Adaptive mesh approach from the class of "mesh moving" methods.
Implementation	Method of Lines
Spatial discretisation	Galerkin Finite Element (with linear trial functions)
Adaptive grid control	Based on minimisation of PDE residuals.
Temporal integrator	Linearly implicit ODE system solver (NAG [39] routine D02NHF)
Jacobian structure	Block-tridiagonal matrix of order 2(NN+2) and band width 7
Method parameters	**d** minimum node separation **C1** inter-nodal viscosity parameter **C2** inter-nodal spring force parameter

Table 6.3: Characteristics of the Moving Finite Element Method.

6.2 Example problems

Versatility has already been established as an important requirement for adaptive mesh strategies. In order to test the versatility of each adaptive mesh algorithm a suitably varied selection of problems must be solved. The four examples given below, most of which have already been introduced in Chapter 2, represent such a selection. The problems present novel difficulties for the numerical methods and the solutions are widely different in nature.

Problem 1

A "hot spot" problem from combustion theory.

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2} + D(1+a-u)\exp(-\frac{\delta}{u}) \qquad \begin{array}{l} 0 \le x \le 1 \\ t \ge 0 \end{array}$$

where D is given by

$$D = \frac{R\exp(\delta)}{a\delta}$$

and the boundary and initial conditions are

$$\begin{aligned} \frac{\partial U}{\partial x}(0,t) &= 0 \quad U(1,t) = 1 \\ U(x,0) &= 1 \end{aligned}$$

An accurate numerical solution is used as a reference solution.

Source: Adjerid and Flaherty [1]

Problem 2

Burgers' equation: sine wave initial conditions. (Problem 7a, Chapter 2)

$$\frac{\partial U}{\partial t} = \epsilon\frac{\partial^2 U}{\partial x^2} - U\frac{\partial U}{\partial x} \qquad \begin{array}{l} 0 \le x \le 1 \\ t \ge 0 \end{array}$$

with initial conditions

$$U(x,0) = \sin(2\pi x) + 0.5\sin(\pi x)$$

and boundary conditions

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

An accurate numerical solution is used as a reference solution.

Source: Gelinas, Dos and Miller [23].

Problem 3

Burgers' equation: propagating shocks. (Problem 7b, Chapter 2)

$$\frac{\partial U}{\partial t} = \epsilon\frac{\partial^2 U}{\partial x^2} - U\frac{\partial U}{\partial x} \qquad \begin{array}{l} 0 \le x \le 1 \\ t \ge 0 \end{array}$$

with initial and boundary conditions

$$\begin{array}{rcll} U(x,0) & = & U_e(x,0) & U(0,t) = U_e(0,t) \\ U(1,t) & = & U_e(1,t) & \end{array}$$

The exact solution $U_e(x,t)$ is given by

$$U_e(x,t) = 1 - 0.9\frac{r_1}{R} - 0.5\frac{r_2}{R}$$

where

$$\begin{array}{rcl} R & = & r_1 + r_2 + r_3 \\ r_1 & = & \exp(-(x - 0.5 + 4.95t)/(20\epsilon)) \\ r_2 & = & \exp(-(x - 0.5 + 0.75t)/(20\epsilon)) \\ r_3 & = & \exp(-(x - 0.375t)/(2\epsilon)) \end{array}$$

Source: Sincovec and Madsen [32].

Problem 4
A model parabolic equation. (Problem 4, Chapter 2)

$$\frac{\partial U}{\partial t} = \sigma \frac{\partial^2 U}{\partial x^2} + f(x) \qquad \begin{array}{l} 0 \le x \le 1 \\ t \ge 0 \end{array}$$

with boundary and initial conditions

$$\begin{aligned} U(0,t) &= \tanh(r_2 t - r_1) \quad U(1,t) = \tanh(r_2 t) \\ U(x,0) &= \tanh(r_1(x-1)) \end{aligned}$$

and exact solution

$$U(x,t) = \tanh(r_1(x-1) + r_2 t)$$

The source term f(x) is given by

$$f(x) = sech^2(r_1(x-1) + r_2 t)[r_2 + 2\sigma \tanh(r_1(x-1) + r_2 t)$$

Source: Davis and Flaherty [14].

6.3 Measures of effectiveness

In the last chapter the three recognised characteristics of effective software for parabolic problems were introduced. These were versatility, robustness and efficiency. The selected adaptive mesh algorithms (Methods II and III) will now be assessed in relation to these criteria. For comparison purposes the conventional fixed grid MOL approach (Method I) will also be included in the assessment.

In the following experiments effectiveness is measured in terms of accuracy and efficiency. The accuracy of each algorithm is illustrated in two ways. Firstly, exact/reference solutions (denoted by solid lines) are plotted along with the numerical solutions (plotted symbols). Secondly, the maximum absolute error between the numerical and exact/reference solutions (over all the specified output times) is quoted in the tables. The efficiency of each algorithm may also be expressed in two ways. Firstly, the CPU time consumption provides an excellent means of determining the relative efficiency of several algorithms, implemented on the same hardware. The actual computer used was an APOLLO workstation (DN4500) and the CPU time consumption, measured in seconds, was determined for each implementation. A second means of expressing the efficiency of the algorithms is to determine operation counts. These statistics are particularly useful since they are characteristic of the particular algorithms rather than the hardware on which they are implemented. In the tables to follow, emphasis is placed on the use of high level operation counts such as Jacobian evaluations, rather than on low level indicators such as multiplications or additions.

6.4 The numerical time integration

The two adaptive mesh strategies (Methods II and III) are similar to Method I in that they are both Method of Lines (MOL) implementations. The spatial discretisation in each case gives rise to linearly implicit ODE systems of the form

$$A(t,y)y' = g(t,y) \qquad \begin{array}{l} y(t_0) = y_0 \\ t_0 \le t \le t_{final} \end{array} \tag{6.1}$$

Here, A is a matrix of order 2(**NN**+2) and y, g are vectors of dimension (**NN**+2), where **NN** is the number of internal spatial mesh points used in the semidiscretisation.

The system (6.1) can be solved numerically using the BD formulae (see Chapter 3 section 3.2.2) where

$$-\beta_0 h y'_{n+1} = -y_{n+1} + \sum_{k=1}^{K} \alpha_k y_{n+1-k}$$

is the K^{th} order BD formula with coefficients $\beta_0, \alpha_1, \alpha_2, \ldots, \alpha_K$ and h is the time step size. This gives

$$\begin{aligned} y_{n+1} &= \sum_{k=1}^{K} \alpha_k y_{n+1-k} + h\beta_0 y'_{n+1} \\ &= a_{n+1} + h\beta_0 y'_{n+1} \end{aligned}$$

Here, a_{n+1} is a constant vector constructed from past values of y. Using this formula to approximate y'_{n+1} in (6.1) gives

$$A(t_{n+1}, y_{n+1})\,(y_{n+1} - a_{n+1}) - h\beta_0 g(t_{n+1}, y_{n+1}) = 0 \tag{6.2}$$

The user supplied residual

$$r(y_{n+1}) = g(t_{n+1}, y_{n+1} - a(t_{n+1}, y_{n+1}) y'^{(0)}_{n=1}$$

is now introduced where $y'^{(0)}_{n+1}$is the predicted value of y'_{n+1} given by

$$y'^{(0)}_{n+1} = \frac{y^{(0)}_{n+1} - a_{n+1}}{h\beta_0}$$

Thus (6.2) can be rewritten as

$$A(t_{n+1}, y_{n+1})\left(y_{n+1} - y^{(0)}_{n+1}\right) - h\beta_0 r(t_{n+1}, y_{n+1}) = 0 \tag{6.3}$$

This nonlinear algebraic system is solved using a modified Newton iteration where the iteration matrix

$$P = A(t_{n+1}, y^{(0)}_{n+1}) - h\beta_0 r'(y^{(0)}_{n+1}) \tag{6.4}$$

approximates the Jacobian of the system (6.3). Because the solution of this system involves iteration, an important statistic is the total number of iterations used in a particular integration. This is denoted by **NITER** in the results to follow. The step size h is automatically set by the BDF codes although an initial step size **H0** can be specified by the user. It is also useful to know the total number of time steps taken in a particular implementation. This is denoted by **STEPS**. The iteration of equation (6.3) differs from a true Newton iteration in that the Jacobian is only evaluated periodically. This occurs after the prediction stage (when $y^{(0)}_{n=1}$ is given) and otherwise when necessary eg. if a convergence test fails. The calculation of the Jacobian (performed numerically) is rather expensive and so a useful statistic to consider is the total number of Jacobian evaluations, denoted by **NJACS**. The number of Jacobian evaluations also corresponds to the number of LU decompositions of the iteration matrix in (6.3). A Newton iteration process is terminated when a Newton correction is found which, in the chosen norm, is less than the user specified tolerance, **TOL**. The last statistic of interest is the number of evaluations of the residual r(y), denoted by **NRE**.

Application of the conventional MOL approach of Method I gives rise to a different ODE problem. Here semidiscretisation results in a system of explicit ODEs of the form

$$y' = g(t, y)$$

This is equivalent to the system (6.1) where the matrix A is replaced by the identity matrix I and the iteration matrix P in (6.4) becomes

$$P = I - h\beta_0 g'(y_{n+1}^{(0)}) \tag{6.5}$$

However in this case the orders of P and A are now (**NN** + 2) because in the conventional Method of Lines the extra N equations, corresponding to the moving grid in the adaptive mesh algorithms, are not present. The statistics **NITER**, **STEPS**, **NJACS** and **NRE** still retain the same meaning, however, except that **NRE** now refers to g(t,y) evaluations rather than to r(t,y) evaluations. In the case of Method I the meaning of this parameter can therefore be interpreted as the "number of right hand side evaluations".

In comparing implementation statistics for each of the three methods, care must be taken to remember the differences between them. The most important difference is that the Jacobian band widths for the three methods are 3,7 and 9, respectively, with the order of the Jacobian in each case being determined by **NN**. These differences will be reflected in the CPU statistics for each method. The ODE integrators used in the three methods are implemented in essentially the same way. For all implementations numerical differencing was used to evaluate Jacobians. This was performed internally by the integrators (on request). A subroutine was provided which performed the residual/function evaluations and the initial solution Y(0) and required output times were also specified. For the local error test in the Newton iteration scalar relative and absolute error tolerances, **RTOL** and **ATOL**, were chosen such that **RTOL** = **ATOL** = **TOL**. For the experiments **TOL** was set to a value of 10^{-3} unless otherwise stated and the chosen norm used in the tolerance test was the common "averaged L_2" norm. For all experiments the initial step size in the time integration, **H0**, was set to 10^{-5}. Lastly, for all the experiments, the start grid used in the adaptive mesh algorithms, was uniform.

6.5 Results

The following sections give the numerical results for the adaptive mesh strategies (Methods II and III) applied to the solution of the four example parabolic problems of section 6.2. For each problem the performance of the adaptive strategies are compared with the conventional fixed grid MOL approach of Method I. A summary of the "figures of merit" used in the tables to follow is now given.

TOL	The scalar relative and absolute error tolerance used in the time integration.
H0	The initial time step size.
NN	The number of internal grid points in the spatial discretisation.
STEPS	The number of time steps used.
NJACS	The number of Jacobian evaluations (=LU decompositions).
NRE	The number of residual or right-hand-side evaluations.
NITER	The total number of Newton iterations.
CPU	The CPU time (in seconds) required for the solution. All experiments were carried out on an APOLLO DN4500 and while determining CPU time all input/output operations were disabled.
ERR	The maximum absolute error on the spatial interval, over all the specified output times.

6.5.1 Problem 1

The solution to Problem 1 represents the temperature of a reactant in a chemical system. For small values of time the temperature at X=0 increases from unity causing a "hot spot" to develop at this point. At a finite time ignition occurs causing the temperature at X=0 to rapidly increase to a value 1 + a. This leads to the formation of a flame front which rapidly propagates towards X=1 at a speed proportional to $\exp(\alpha\delta)/2(1+\alpha)$.

Following [1,53] the parameters of the equation are chosen as follows. [a=1, d=20, R=5]. The problem reaches a steady state when the flame front reaches the right boundary. For the present choice of parameters this occurs at T≈0.29. In the numerical solution two distinct phases can be identified, viz., the ignition phase (formation of the hot spot at X=0) and the propagation phase (movement of the flame front towards X=1).

Simulating the ignition phase is difficult because ignition occurs very rapidly. This small time-scale phenomenon causes considerable stiffness in the numerical solution, necessitating the use of variable time steps. The start of the ignition must be accurately determined without overshoot in the local error control mechanism of the stiff ODE solver. This allows sufficiently small time steps to be used in order to effectively simulate the ignition. Errors at this stage result in considerably greater errors later on. Following [1,53] a time tolerance of approximately **TOL**=10^{-5} (with an initial time step size of **H0**=10^{-5}) was chosen in order to effectively capture the ignition phase. Such a small tolerance does not seriously affect the efficiency of the BDF codes since they are capable of stepping in time using higher order formulae.

The presence of the ignition stage in this problem makes the time stepping process much more difficult than the spatial discretisation. Indeed, the flame front is not particularly thin and can be satisfactorily resolved using a conventional fixed spatial grid with as little as 40 internal nodes. Figure 6.1 shows the the numerical solution of the problem (plotted symbols) computed using the fixed grid MOL approach of Method I, for various values of **NN**. An accurate reference solution is also plotted (solid lines) for comparison purposes. The reference solution was generated using Method I with **NN**=2000, **TOL**=10^{-8} and with **H0**=10^{-5}. The integration statistics corresponding to these solutions are shown in Table 6.4. Note the characteristic efficiency of the conventional MOL approach in that the number of Jacobian evaluations, **NJACS**, is always a small fraction of the number of time steps, **STEPS**. The operation counts for **NN**=40 are a lot less than for **NN**=10 but in fact the computational cost is much higher since, in the former case, the order of the Jacobian is approximately four times as great. This is clearly revealed by the CPU statistics, which steadily increase with increasing **NN**. The global error, **ERR**, starts off relatively large for **NN**=10 but decreases rapidly as **NN** increases. For **NN**=30,40 the value of **ERR** appears to stabilise at approximately 0.05. This is because of the large difference between the reference and numerical solutions at T=0.26, X=0, visible in Figure 6.1. This discrepancy arises because the integrator has difficulty in determining the start of the ignition phase. A smaller value of **TOL** would have improved the performance considerably. For the present value of **TOL**, however, the integration proceeds well at later times and, as can be seen from Figure 6.1, the reference and numerical solutions are almost indistinguishable at these times.

For the adaptive mesh strategies the present problem is of interest because the Lagrangian nature of these methods allow the grid to track the propagating flame front, thereby reducing the stiffness of the problem and allowing fewer time steps to be used during the propagation phase. Figures 6.2 and 6.3 show the grid trajectories and solutions at four different output times using Method II (with **NN**=40), for various values of the method parameter τ. For the case τ=1.0, a uniform fixed grid implementation results, with the grid becoming increasingly adaptive as τ decreases. It should be remembered, however, that a fixed grid implementation of Method II with **NN**=40 is not equivalent to the corresponding implementation of Method I (also with **NN**=40), since in the former case the order of the Jacobian is approximately twice as great.

Table 6.5 shows the integration statistics for the solutions obtained with Method II. For the smallest τ value (=0.0001), corresponding to the most adaptive grid, the number of time steps used, **STEPS**, is almost half that used for the case τ=1.0, (uniform, fixed grid). This demonstrates the desired effect of the Lagrangian approach of Method II. As in the solutions obtained with Method I, the global error is exclusively dominated by the large discrepancy between the reference and numerical solutions at T=0.26, X=0. At later times, however, the numerical solutions are not quite as accurate as those obtained with Method I. This is because Method II tends to overestimate the speed of the flame front during the propagation phase. In Table 6.5, as τ decreases the various operation counts also decrease with the result that CPU reduces to a value of 61 seconds for the smallest τ value, compared with a value of 122 seconds for the largest τ value. The greater efficiency of the variable grid implementation of Method II, over the fixed grid alternative, is obvious.

The present problem was also solved using Method III with the parameters **C2** and **d** set to zero. Varying the value of **C1** has the same effect as varying τ in Method II. This parameter is associated with the "inter-nodal viscosities" discussed in the last chapter. Figure 6.4 shows the grid trajectories and solutions computed using Method III for two values of **C1**. The value **C1**=10.0 corresponds to a uniform fixed grid implementation. Table 6.5 lists the integration statistics for these implementations and also for two other implementations. For the moving grid situations the method gives consistently better approximations of the flame speed (indicated by smaller **ERR** values) compared with Method II. Notice in Figure 6.4 that the errors at T=0.26, X=0 are considerable less than those obtained using Methods I and II. The higher accuracy of Method III over Method II comes at a considerably greater cost, however, as indicated by the consistently higher values of **CPU** for Method III. Unlike Method II the operation counts do not continuously decrease with a reduction in **C1**. Indeed, for most implementations, too small a value of **C1** may result in node overtaking. This suggests that the choice of **C1** in Method III is somewhat more difficult than the corresponding choice of τ in Method II which is a disadvantage with respect to robustness and ease of implementation.

For this problem, the results suggest that the implementations of Methods II and III are more effective when used in "adaptive grid" mode rather than "fixed grid" mode. However the viability of these two methods can only be properly assessed when compared with the standard fixed grid MOL approach of Method I. The comparison of new adaptive mesh strategies with the conventional approaches is an exercise which has been overlooked in the literature on adaptive meshing. Comparing the most accurate results of Method II and III with the corresponding results of Method I (for **NN**=40) gives the following statistics.

	STEPS	**NJACS**	**NITER**	**CPU**	**ERR**
Method I	376	39	500	18.5	0.06
Method II	204	39	348	61.0	0.07
Method III	350	195	1023	179.5	0.04

Obviously, the desired effect of the Lagrangian approaches of Methods II and III have been realised in that **STEPS** is less for these methods. However, the values of **CPU** reveal that Method I is between three and ten times more efficient than Methods II and III. Clearly, for this problem, the conventional fixed grid MOL approach of Method I is far superior. On the whole, this problem is not a good demonstration of the superiority of adaptive meshing over the conventional approach but it does illustrate vividly

Problem 1

a = 1 **d** = 20 R = 5

Method I

Method parameters

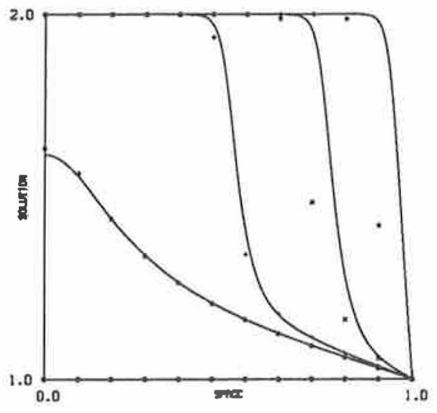
TOL	**H0**
10^{-5}	10^{-5}

Integration statistics

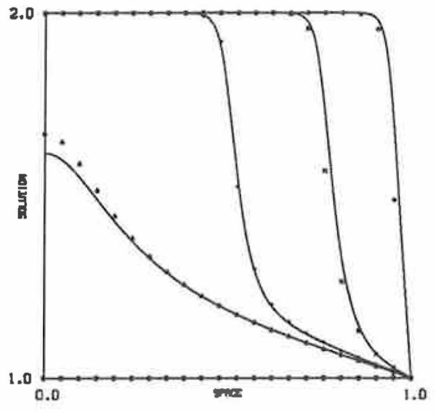
	NN = 10	NN = 20	NN = 30	NN = 40
STEPS	431	451	428	376
NJACS	68	53	44	39
NRE	852	806	712	619
NITER	646	645	578	500
CPU	7.3	12.7	16.0	18.5
ERR	0.54	0.17	0.05	0.06

Table 6.4: Integration statistics for Problem 1 solved using Method I.

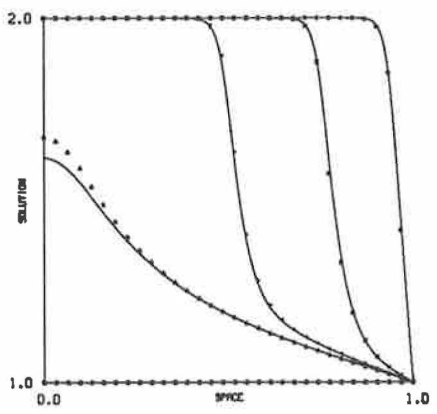
the Lagrangian nature of Methods II and III.



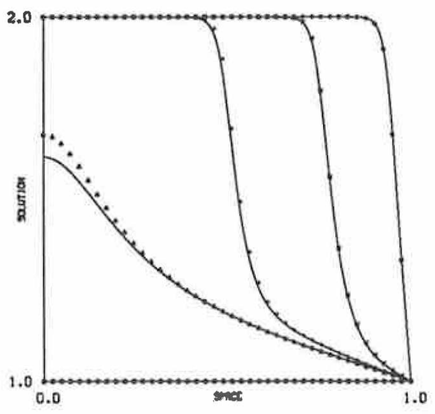
(a)



(b)

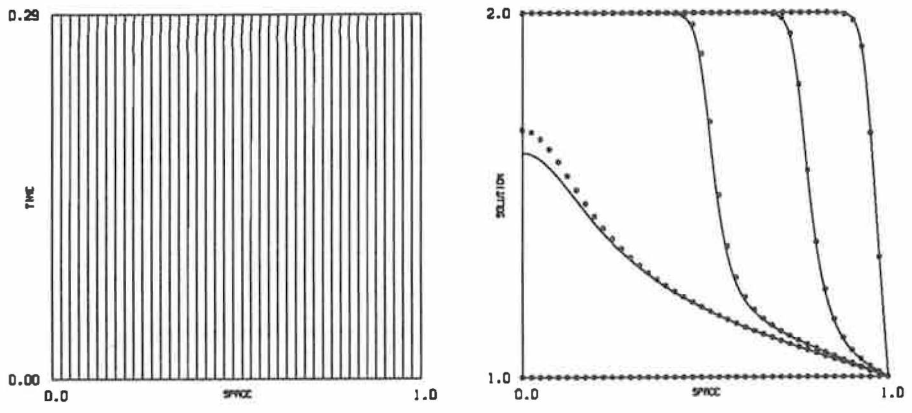


(c)

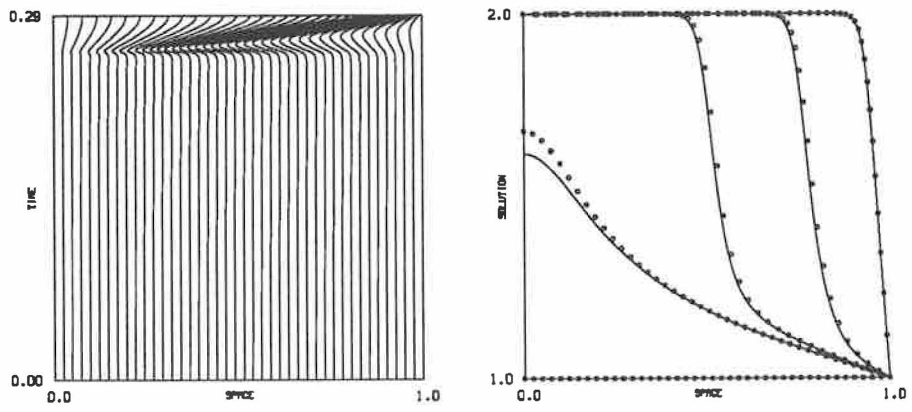


(d)

Figure 6.1: Solutions for Problem 1 generated using Method I for the output times T=0.26, 0.27, 0.28, 0.29 using (a) NN=10, (b) NN=20, (c) NN=30, (d) NN=40

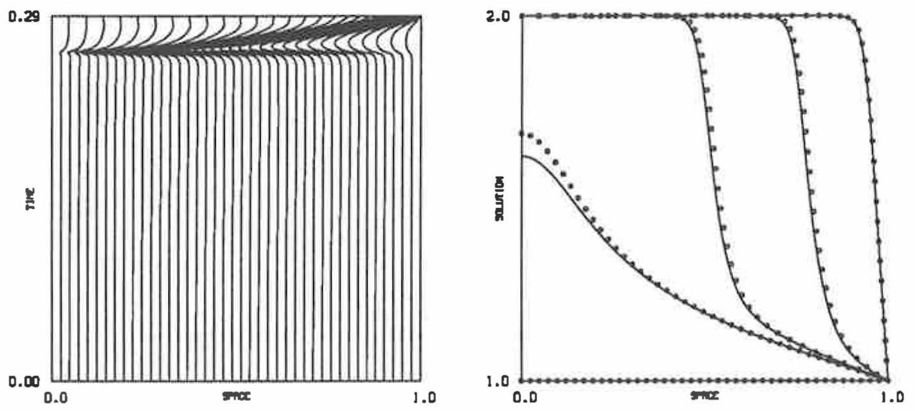


(a)

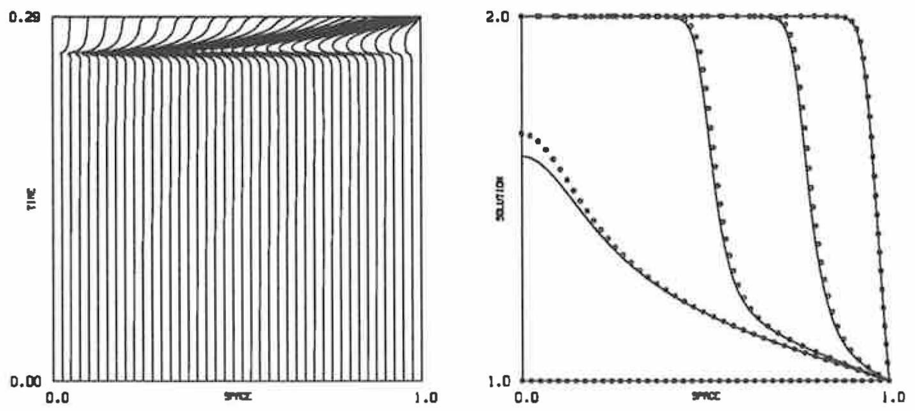


(b)

Figure 6.2: Solutions and grid trajectories for Problem 1 generated using Method II for the output times T=0.26 0.27 0.28 0.29 with **NN**=40 and (a) τ=1.0, (b) τ=0.01

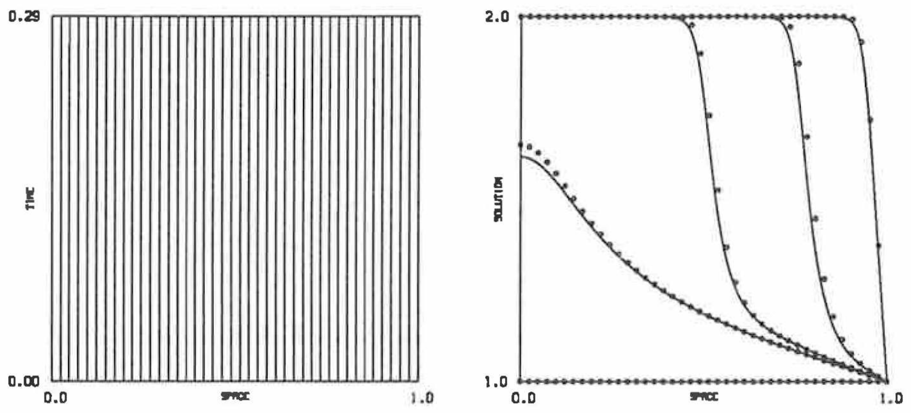


(a)

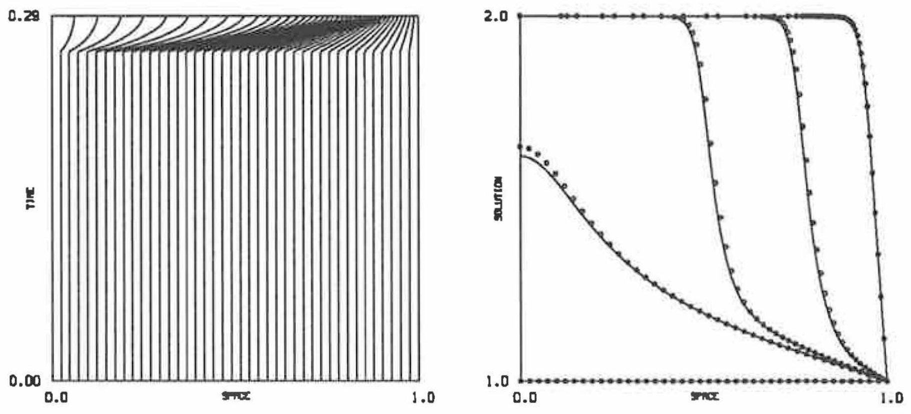


(b)

Figure 6.3: Solutions and grid trajectories for Problem 1 generated using Method II for the output times T=0.26 0.27 0.28 0.29 with NN=40 and (a) τ=0.001, (b) τ=0.0001



(a)



(b)

Figure 6.4: Solutions and grid trajectories for Problem 1 generated using Method III for the output times T=0.26, 0.27, 0.28, 0.29 with **NN**=40, **C2**=**d**=0.0 and (a) **C1**=10.0, (b) **C1**=0.05

Problem 1

a = 1 **d** = 20 R = 5

Method II

Method parameters

κ	α	**TOL**	**H0**	**NN**
2	1	10^{-5}	10^{-5}	40

Integration statistics

	$\tau = 1$	$\tau = 0.01$	$\tau = 0.001$	$\tau = 0.0001$
STEPS	439	277	215	204
NJACS	79	50	42	39
NRE	1411	904	764	712
NITER	679	439	373	348
CPU	122.3	78.9	65.2	61.0
ERR	0.06	0.07	0.08	0.07

Method III

Method parameters

C2	**d**	**TOL**	**H0**	**NN**
0	0	10^{-5}	10^{-5}	40

Integration statistics

	C1 = 10	**C1** = 0.1	**C1** = 0.05	**C1** = 0.025
STEPS	352	311	326	350
NJACS	51	141	165	195
NRE	1020	1911	2133	2435
NITER	604	889	937	1023
CPU	88.4	144.0	151.0	179.5
ERR	0.12	0.06	0.08	0.04

Table 6.5: Integration statistics for Problem 1 solved using Method II and III with NN=40.

6.5.2 Problem 2

The initial conditions for this problem are quite smooth but the solution is a wave which steepens rapidly and then propagates towards X=1. The presence of homogeneous Dirichlet boundary conditions means that the solution dampens towards an asymptotic value of zero at later times. Conventional fixed grid MOL solutions, based on centered Finite Differences, give rise to spurious oscillations when the grid is too coarse in the steep region of the solution. For the moving mesh approaches the placement of mesh points in this region is critical, with slight deviations from the optimal grid causing oscillations. The resulting non-smoothness (wriggles), induced in the solution, can seriously affect the efficiency of the stiff ODE solver. The presence of a steep moving front means that the spatial discretisation is more difficult than the time stepping process. For this reason the tolerance in the BDF codes (for all the methods) was set to **TOL**=10^{-3}; relatively large compared to the value chosen in the previous problem. In the numerical results to follow the reference solution was calculated using Method I with **NN**=2000, **TOL**=10^{-8} and **H0**=10^{-5}.

Figure 6.5 shows the results obtained with Method I at the output times T=0.2, 0.6, 1.0, 1.4, 2.0 for two choices of **NN**. Method I failed repeatedly when **NN** was less than about 500. This was because the steep wavefront could not be resolved. Even for **NN**=500 there are hardly any nodes present within the steep front and some oscillations can be seen at the crest of the wave. For **NN**=600 the solution is adequately resolved, with several nodes being present in the shock front. Table 6.6 shows the integration statistics for both implementations of Method I. The MOL approach does a good job at keeping **NJACS** a small fraction of **STEPS**. Both implementations give effective results with an increase in **NN** causing a sharp decrease in **ERR**. As indicated by the CPU statistics, Method I behaves very inefficiently with the solution for **NN**=600 requiring a **CPU** value of nearly nine minutes. This is due to the large order Jacobian arising from the spatial discretisation.

Adaptive mesh solutions to this problem are shown in Figures 6.6 and 6.7 using Methods II and III, respectively. In both cases **NN**=40. Notice the sharp movement of the grid trajectories at $T \approx 1.3$. This is caused by a sudden change in the shape of the solution which must be adequately resolved by the mesh moving methods. Figure 6.6 shows inaccurate results for Method II for the times T=1.2, 1.4, whereas both implementations of

Method III are much more accurate at all the specified output times. The surprisingly large **ERR** values quoted for Method III in Table 6.7 arise due to slight deviations between the numerical and reference shock positions. Because the shock is very steep these relatively small lateral deviations give rise to relatively large absolute errors. The solutions delivered by Method III remain effective with the numerical and reference solutions being almost indistinguishable in Figure 6.7. Table 6.7 indicates typical behaviour for Method II in that as τ decreases the efficiency improves. As was the case in the previous problem the correct choice of parameters for Method III is not obvious and the costs for various values of **C1**, **C2** and **d** fluctuate considerably.

C1 and **C2** are associated with the inter-nodal "viscosity" and "spring" functions of the Moving Finite Element method, respectively.. **C1** is used to avoid parallelism of the moving nodes while **C2** avoids node overtaking. The parameter **d** is a prescribed minimum node separation which should be smaller than the expected small-scale structure in the solution. Thus, for the present problem, **d** must be less than the value of the diffusion coefficient, ϵ, since the width of the wave front is proportional to this value. Unfortunately the use of such a minimum node separation ultimately means that the optimal choice for **d** is essentially problem dependent. From the CPU statistics given in Table 6.7, Method III is evidently a lot less efficient than Method II but even for such a wide choice of method parameters the results are always accurate. Method III succeeds in remaining robust but at the expense of efficiency.

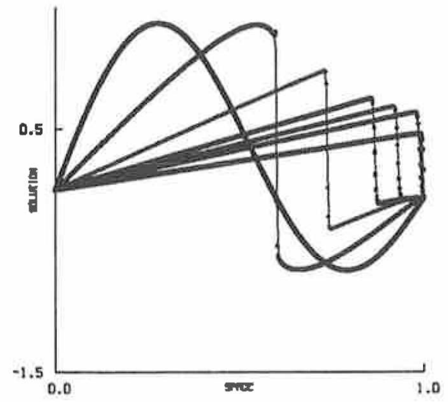
For this problem results for Method II were unsatisfactory as regards robustness and, for smaller values of τ than those given in Table 6.7, fatal Newton errors were repeatedly encountered when the minimum mesh spacing became very small. Similar problems were reported by Verwer et al. [53] although only for more difficult problems than the present example. The reason why they appear more often in the present experiments is because the implicit ODE solver used here is the NAG routine D02NHF [39] whereas the solver used by Verwer et al. is the more advanced DASSL code [43]. This package is especially superior in its treatment of Differential/Algebraic systems which occur in Method II when τ is set to zero. In all experiments very small τ values were avoided in order to avoid the possible shortcomings of the ODE solver being used.

Comparing the results obtained with Method III in Figure 6.7(b) (NN=40) with those obtained using Method I in Figure 6.5(b) (NN=600) gives the following statistics.

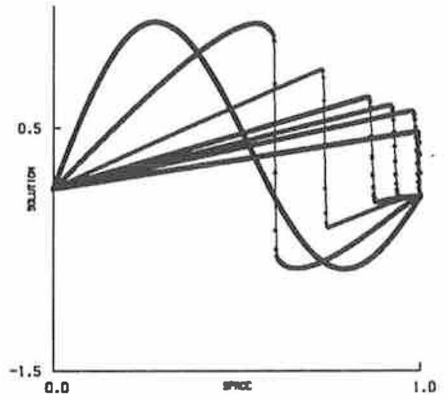
	STEPS	**NJACS**	**NITER**	**CPU**	**ERR**
Method I	780	61	1023	534.0	0.08
Method III	208	163	652	121.0	0.41

Although not nearly as accurate as Method I, Method III performs the solution in one quarter of the time required by Method I. Also, as can be seen from Figures 6.7(b) and 6.5(b) the solutions are qualitatively not very different.

Overall, this problem illustrates the effectiveness of the adaptive mesh algorithms very well, especially for problems where the spatial resolution is paramount. Method III works robustly and accurately but the difficulty of choosing optimal values of the method parameters **C1**, **C2** and **d** still remains. This leads to implementations which are non-optimal. For a difficult problem, such as the present example, where the standard approach of Method I fails repeatedly, Method III is very much a viable alternative.



(a)



(b)

Figure 6.5: Solutions of Problem 2 generated using Method I for the output times T=0.2, 0.6, 1.0, 1.2, 1.4, 2.0 (a) NN=500, (b) NN=600

Problem 2

$\epsilon = 0.001$

Method I

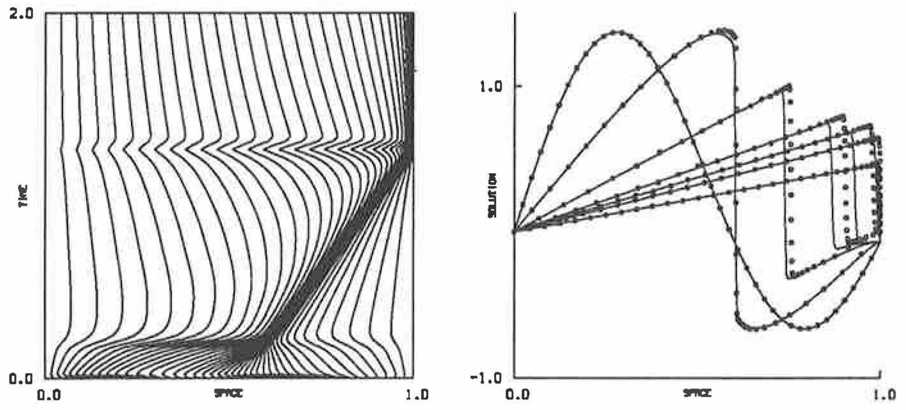
Method parameters

TOL	**H0**
10^{-3}	10^{-5}

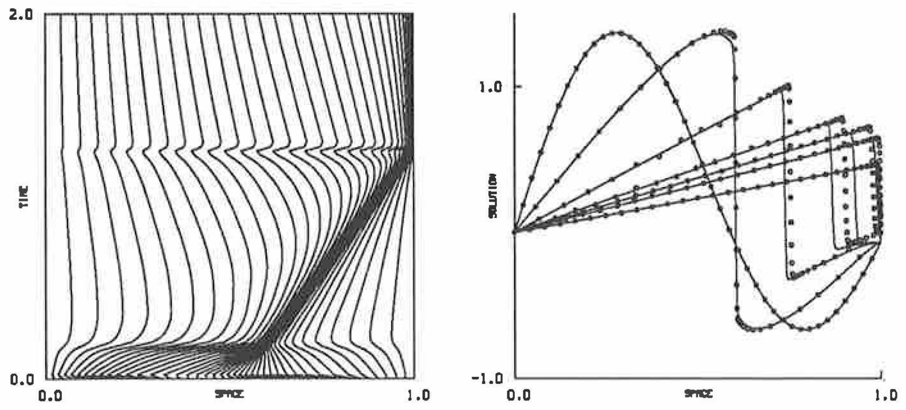
Integration statistics

	NN = 500	**NN** = 600
STEPS	831	780
NJACS	74	61
NRE	1299	1208
NITER	1075	1023
CPU	459.5	534.0
ERR	0.13	0.08

Table 6.6: Integration statistics for Problem 2 solved using Method I.

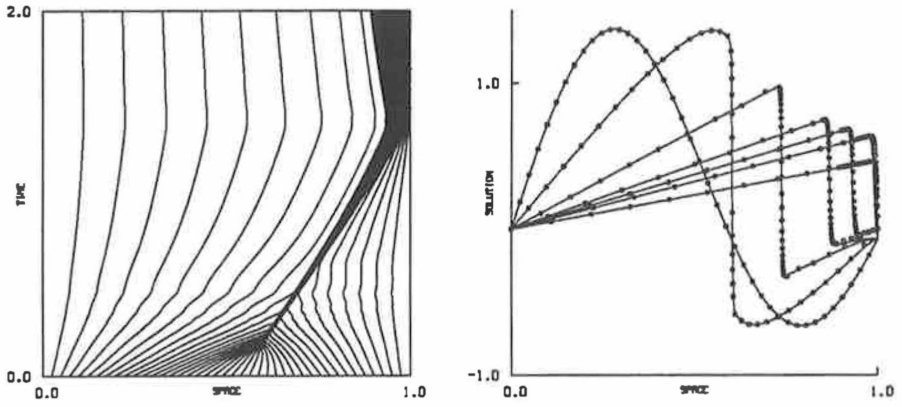


(a)

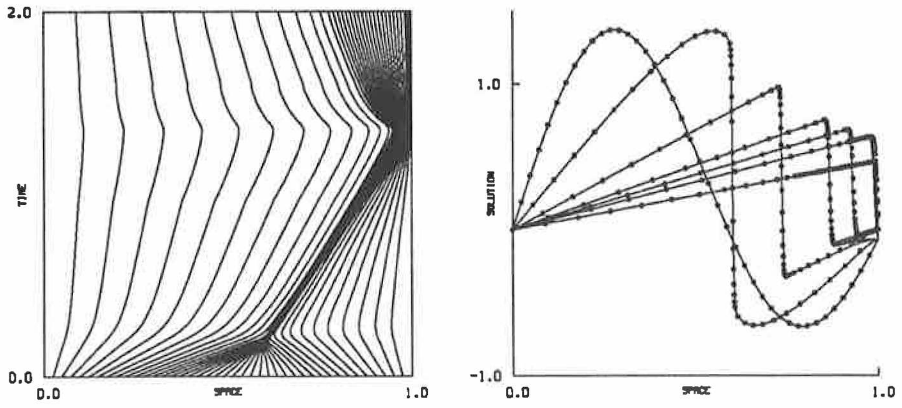


(b)

Figure 6.6: Solutions and grid trajectories for Problem 2 generated using Method II for the output times T=0.2, 0.6, 1.0, 1.2, 1.4, 2.0 with NN=40 and (a) τ=0.01, (b) τ=0.001



(a)



(b)

Figure 6.7: Solutions and grid trajectories for Problem 2 generated using Method III for the output times T=0.2, 0.6, 1.0, 1.2, 1.4, 2.0 with **NN**=40 and (a) **C1**=0.025, **C2**=0.0001, **d**=0.0001 and (b) **C1**=0.01, **C2**=0.0001, **d**=0.0005

Problem 2 ($\epsilon = 0.001$)

Method II

Method parameters

κ	α	**TOL**	**H0**	**NN**
2	1	10^{-3}	10^{-5}	40

Integration statistics

	$\tau = 0.01$	$\tau = 0.001$
STEPS	192	157
NJACS	74	63
NRE	1095	932
NITER	404	346
CPU	88.0	71.2
ERR	1.32	1.33

Method III

Method parameters

C2	**TOL**	**H0**	**NN**
0.0001	10^{-3}	10^{-5}	40

Integration statistics

	C1 = 0.025 **d** = 0.0001	**C1** = 0.025 **d** = 0.0005	**C1** = 0.01 **d** = 0.0001	**C1** = 0.01 **d** = 0.0005
STEPS	473	240	439	208
NJACS	342	186	349	163
NRE	3735	2054	3763	1834
NITER	1260	705	1239	652
CPU	251.8	135.6	250.4	121.0
ERR	0.42	0.44	0.43	0.41

Table 6.7: Integration statistics for Problem 2 solved using Methods II and III.

6.5.3 Problem 3

The solution to this problem represents one shock wave overtaking another which then develops into an asymptotic boundary layer at X=1. Both shock structures are clearly visible in the initial solution which is rather non-smooth. This non-smoothness poses problems for both fixed and moving grid approaches. If a uniform start grid is used then a large number of nodes are required in order to accurately resolve the initial solution.

Figure 6.8 shows the solutions obtained using Method I for various values of **NN** with the corresponding integration statistics shown in Table 6.8. The typical oscillations associated with the conventional fixed grid MOL approach occur for **NN**=40,80 and even the solution for **NN**=160 is inaccurate at the tip of the shock (**ERR**=0.03). Only for **NN**=320 do we see a very accurate representation of the shocks and the eventual boundary layer. These results are similar to those obtained in Chapter 3 in the section on "uniform mesh implementations" where this problem was solved using the NAG routine D02PGF [39].

Figures 6.9 and 6.10 show the corresponding solutions obtained with Method II and III with **NN**=40. Notice the rapid movement of the grid in Figure 6.9 (b) at the start of the integration. This occurs because the smaller τ=0.001 value used here, allows the initial grid to rapidly adapt to the form of the initial solution. However, the smoother grid movement obtained with τ=0.1 gives more accurate and efficient results, as seen in Table 6.9. This nicely illustrates the beneficial effect of the temporal grid smoothing property of the method. As in previous problems, however, Method II tends to overestimate the speed of the shocks giving large errors at later times (**ERR**=0.46, 0.68).

As in the last example Method III gives consistently more accurate results than Method II but still remains rather expensive. For the particular parameter choices **C1**=0.025 and C2=0.0001, Table 6.9 shows the effect of varying the minimum node separation, **d**. The larger value of **d** leads to a significant reduction in cost. Ultimately the optimal choice of **d** is somewhat problem dependent and for this particular problem it must not exceed the approximate shock thickness (determined by the diffusion coefficient, ϵ, in Problem 3). This is a negative aspect of the method with respect to robustness. Generally, however, for small values of **d** the method behaves very accurately and robustly, albeit at rather high costs. Comparing the most accurate results of Method III with those of Method I (for **NN**=320) gives the following statistics.

	STEPS	**NJACS**	**NITER**	**CPU**	**ERR**
Method I	191	12	232	60.9	0.01
Method III	85	68	285	52.3	0.011

Method III performs the integration approximately twenty percent faster than Method I while yielding the same global error (**ERR**=0.01).

These results suggest that Method III is more efficient than Method I, although not significantly so. Method II, on the other hand, is approximately twice as fast as Method I but unfortunately rather inaccurate.

Problem 3

$\epsilon = 0.003$

Method I

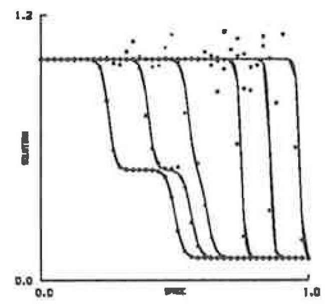
Method parameters

TOL	**H0**
10^{-3}	10^{-5}

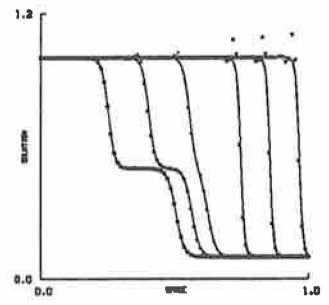
Integration statistics

	NN = 40	NN = 80	NN = 160	NN = 320
STEPS	118	159	165	191
NJACS	9	9	10	12
NRE	167	208	232	270
NITER	138	179	200	232
CPU	5.3	12.7	26.4	60.9
ERR	0.44	0.11	0.03	0.01

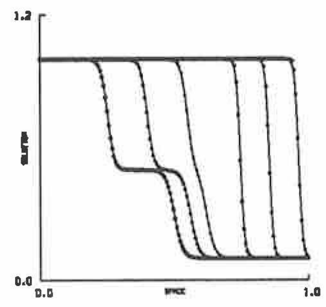
Table 6.8: Integration statistics for Problem 3 solved using Method I.



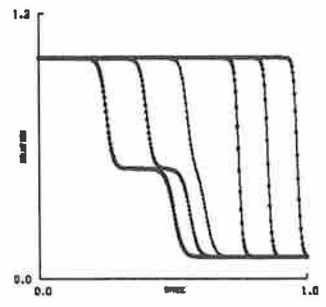
(a)



(b)

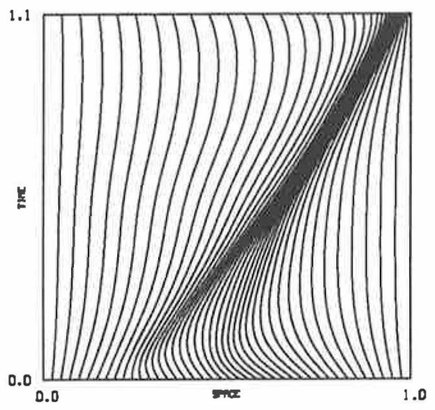


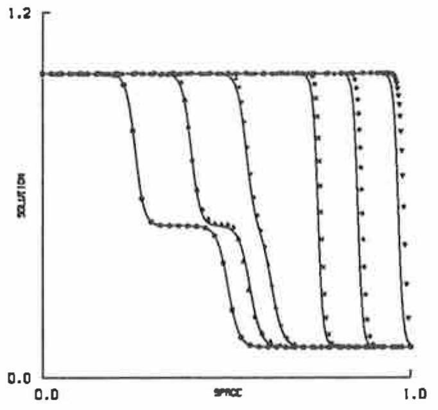
(c)



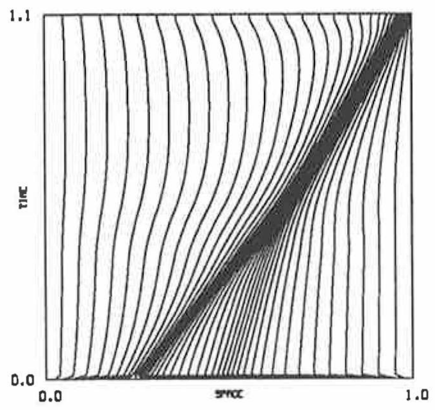
(d)

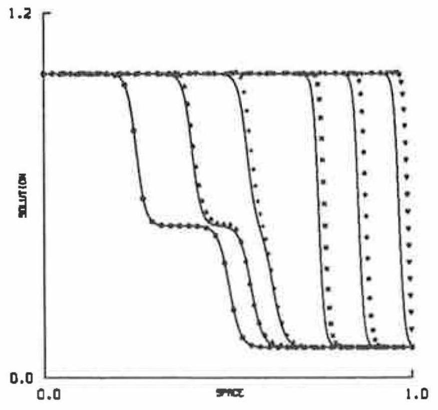
Figure 6.8: Solutions for Problem 3 generated using Method I for the output times T=0.2, 0.4, 0.7, 0.9, 1.1 with (a) NN=40, (b) NN=80, (c) NN=160, (d) NN=320





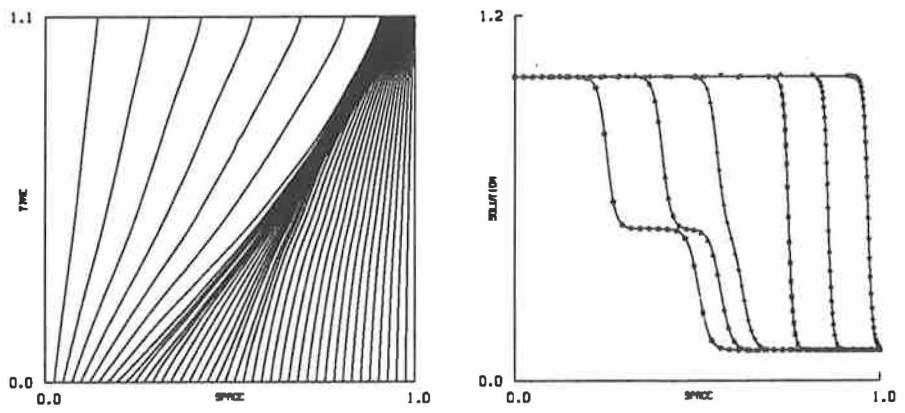
(a)



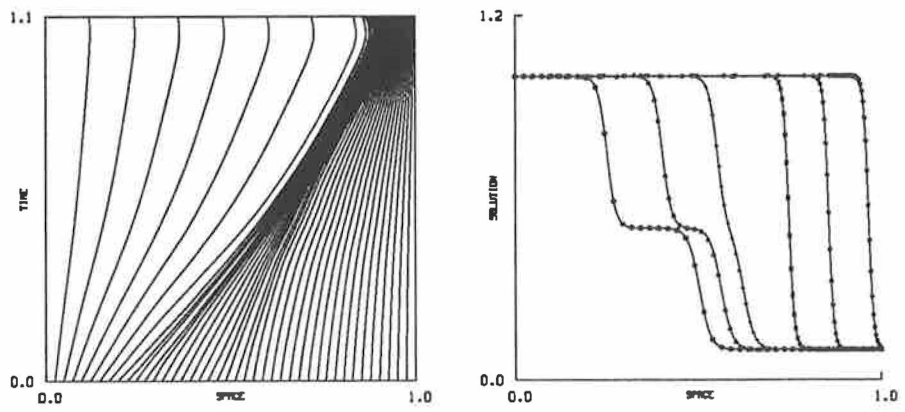


(b)

Figure 6.9: Solutions and grid trajectories for Problem 3 generated using Method II for the output times T=0.2, 0.4, 0.7, 0.9, 1.1 with NN=40 and (a) τ=0.1, (b) τ=0.001



(a)



(b)

Figure 6.10: Solutions and grid trajectories for Problem 3 generated using Method III for the output times T=0.2, 0.4, 0.7, 0.9, 1.1 with **NN**=40, **C1**=0.025, **C2**=0.0001 and (a) **d**=0.001, (b) **d**=0.003

Problem 3 ($\epsilon = 0.003$)

Method II

Method parameters

κ	α	**TOL**	**H0**	**NN**
2	1	10^{-3}	10^{-5}	40

Integration statistics

	$\tau = 0.1$	$\tau = 0.001$
STEPS	77	93
NJACS	26	30
NRE	404	469
NITER	161	188
CPU	34.5	39.8
ERR	0.46	0.68

Method III

Method parameters

C2	**TOL**	**H0**	**NN**
0.0001	10^{-3}	10^{-5}	40

Integration statistics

	C1 = 0.025	**C1** = 0.025
	d = 0.001	**d** = 0.003
STEPS	104	85
NJACS	82	68
NRE	914	780
NITER	317	285
CPU	61.5	52.3
ERR	0.012	0.011

Table 6.9: Integration statistics for Problem 3 solved using Method II and III.

6.5.4 Problem 4

The solution of this problem is a wave which travels in the negative X-direction (when r_1 and r_2 are positive). The steepness and propagation speed of the wave are determined by r_1 and r_2. Following [14] the parameters are chosen as follows [r_1=r_2=100 and δ=10^{-2}]. As in the case of Problem 3, the initial conditions here are quite non-smooth. This causes problems for both the fixed grid and the adaptive mesh implementations.

Figure 6.11 shows the solutions obtained using Method I (for several values of **NN**) and Table 6.10 gives the corresponding integration statistics. For **NN**=20, 40 the results are clearly inadequate (**ERR**≈0.2). For **NN**=80 there is a substantial improvement (**ERR**≈0.1), although the resolution of the solution at the left hand boundary (for T=1.0) is not good. Only for **NN**=160 is the solution accurate over the whole domain (**ERR**=0.03). The solutions obtained with Methods II and III are shown in Figures 6.12 and 6.13 for the case **NN**=40. In Figure 6.12 (a) a large oscillation is present in the solution at T=0.25, 0.5. This is a result of the inadequate node movement allowed by the particular choice of τ(=0.1). For τ=0.001 (Figure 6.12 (b)) the grid adapts quickly to the initial conditions and the overall result is more accurate. In this case **ERR** is the same as that obtained with Method I for **NN**=160.

The results for Method III for two different choices of the parameter **C1** are shown in Figure 6.13. For the present problem, Method III behaves very poorly, giving wild oscillations and large **ERR** values. This is probably because the inner product, associated with the nonlinear term, F(X), of the problem, is inadequately resolved by the Finite Element method. The rather simple *trapezoidal* quadrature rule, used in the formulation, is clearly not accurate enough to handle the F(X) term.

Comparing the most accurate results of Method I and Method II gives the following statistics.

	STEPS	**NJACS**	**NITER**	**CPU**	**ERR**
Method I	644	34	711	113.8	0.03
Method II	194	79	388	61.6	0.03

Clearly, for the same global error, Method II operates nearly twice as efficiently as Method I. Method III, on the other hand, performs twice as slowly but with very poor accuracy.

Problem 4

$\sigma = 10^{-2}$ $R_1 = R_2 = 100$

Method I

Method parameters

TOL	H0
10^{-3}	10^{-5}

Integration statistics

	NN = 20	NN = 40	NN = 80	NN = 160
STEPS	708	909	875	644
NJACS	155	167	86	34
NRE	1584	1882	1343	815
NITER	1117	1379	1083	711
CPU	25.0	56.9	85.9	113.8
ERR	0.20	0.21	0.12	0.03

Table 6.10: Integration statistics for Problem 4 solved using Method I.

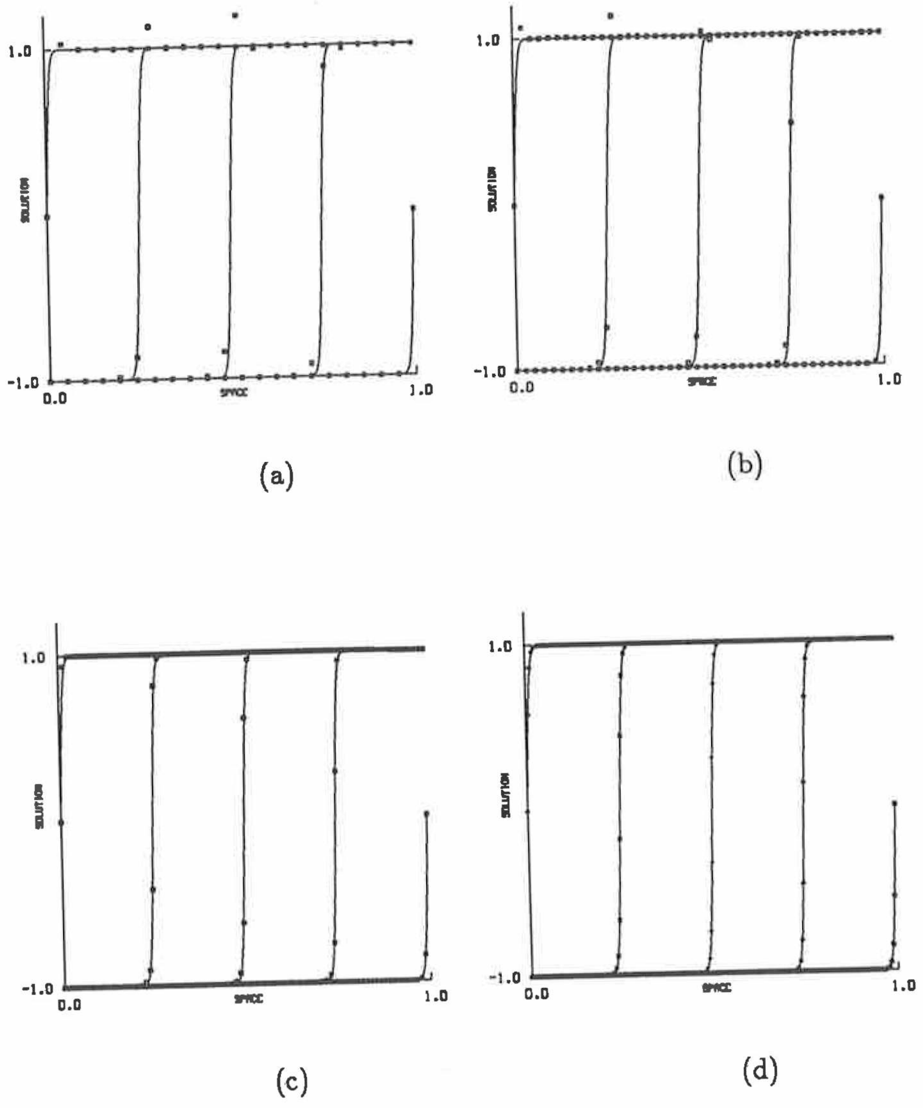
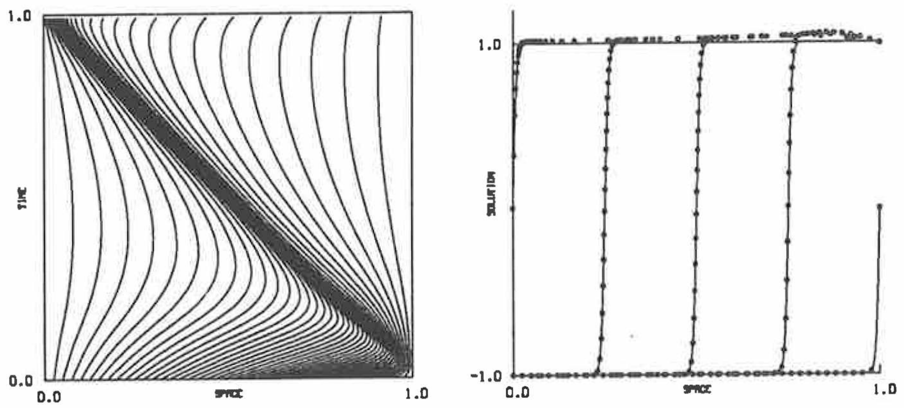
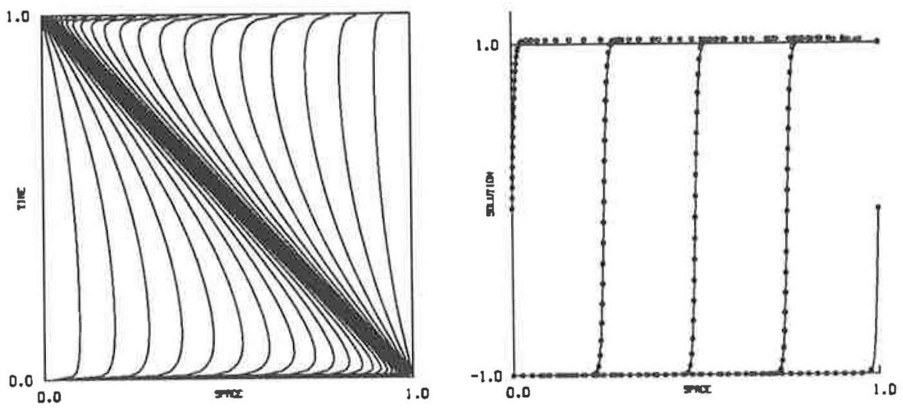


Figure 6.11: Solutions of Problem 4 generated using Method I for the output times T=0.25, 0.50, 0.75, 1.0 using (a) NN=20, (b) NN=40, (c) NN=80, (d) NN=160

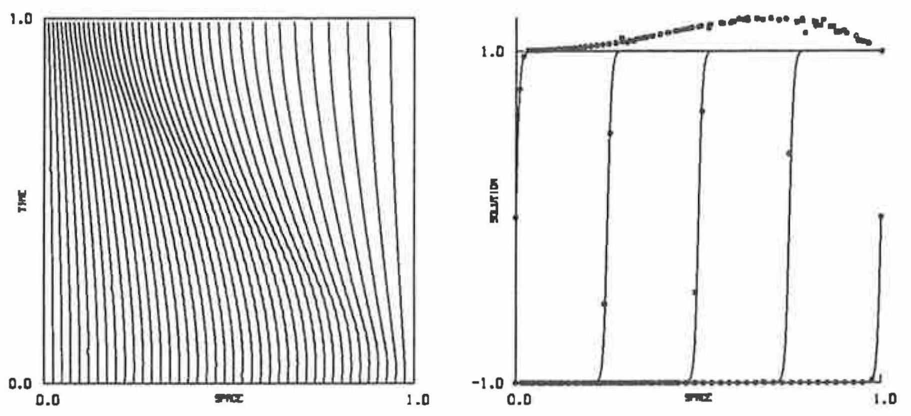


(a)

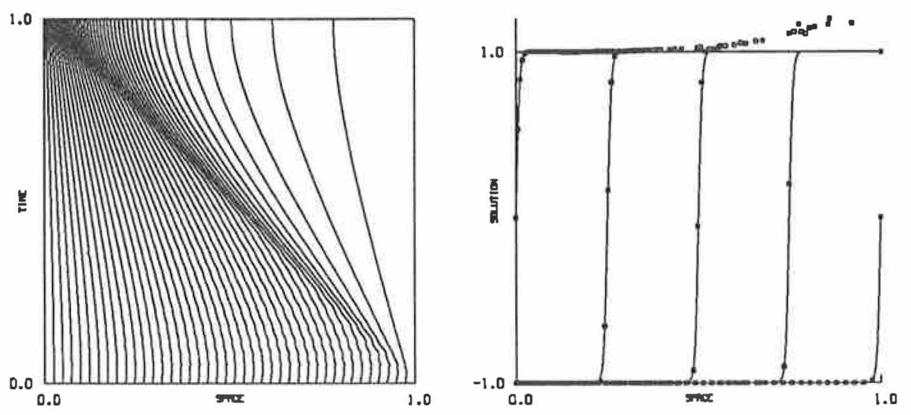


(b)

Figure 6.12: Solutions and grid trajectories for Problem 4 generated using Method II for the output times T=0.2, 0.4, 0.7, 0.9, 1.1 with NN=40 and (a) τ=0.1, (b) τ=0.001



(a)



(b)

Figure 6.13: Solutions and grid trajectories for Problem 4 generated using Method III for the output times T=0.2, 0.4, 0.7, 0.9, 1.1 with **NN**=40, **C2**=**d**=0.001 and (a) **C1**=2.0, (b) **C1**=1.0

Problem 4

$\sigma = 10^{-2}$ $R_1 = R_2 = 100$

Method II

Method parameters

κ	α	**TOL**	**H0**	**NN**
2	1	10^{-3}	10^{-5}	40

Integration statistics

	$\tau = 0.1$	$\tau = 0.001$
STEPS	357	194
NJACS	102	79
NRE	1597	1128
NITER	652	388
CPU	110.8	61.6
ERR	0.05	0.03

Method III

Method parameters

C2	**d**	**TOL**	**H0**	**NN**
0.001	0.001	10^{-3}	10^{-5}	40

Integration statistics

	C1 = 2.0	**C1** = 1.0
STEPS	469	644
NJACS	178	144
NRE	2412	2315
NITER	1123	1270
CPU	238.0	235.0
ERR	0.49	0.26

Table 6.11: Integration statistics for Problem 4 solved using Method II and III.

6.6 Observations

The four parabolic equations of the last section provide a varied set of problem situations which must be handled effectively by the adaptive mesh strategies.

Interestingly, the Finite Difference nature of Method II make it as easy to implement as the standard MOL approach whereas Method III, which is based on Finite Elements, requires somewhat greater effort. Problem 4, which contained a non-linear source term, was solved routinely using Method II since the source term, F(X), could be explicitly incorporated into the formulation. For Method III an inner product involving F(X) had first to be approximated using a quadrature rule. In this case the use of the simple trapezoidal rule was apparently not accurate enough. Generally, Finite Element methods cannot be implemented as automatically as Finite Difference methods owing to the occurrence of these problem-specific inner products.

Both adaptive mesh strategies are quite robust and although neither methods are free of so-called "tuning parameters," a sensible default choice always succeeded in giving reasonable results. The parameter τ in Method II has an obvious meaning in that it is directly associated with the temporal grid smoothing property of the method. Since it is virtually the only parameter required for the adaptive mesh strategy (the other two parameters α and κ were simply set to the values 1 and 2, respectively, in all the numerical experiments) the implementation retains a high level of simplicity.

Method III requires the specification of three parameters **C1**, **C2** and **d**, all of which, except the minimum node separation **d**, are rather indirectly associated with the properties of the mesh. Although the use of these parameters complicates the implementation of Method III considerably, for almost all the choices made in the numerical experiments the method remained very accurate and robust. A favourable property of Method III is that these parameters appear to effect only the cost of the implementation. This is the preferred situation when dealing with tuning parameters; their values should only enhance the existing efficiency of the method rather than critically effect its performance. Regrettably, for Method III very high efficiency was not achieved because of the difficulty in choosing optimal parameter values.

For the implementation of Method II the choice of the parameter τ proved to be somewhat critical to the performance of the method. In the case of Problem 2, for small τ values failures were recorded as a result of the minimum mesh spacing becoming too small. This problem was also allured to by Verwer et al. [53] in similar implementations. However they arose more frequently in the present experiments because the particular DAE/ODE solver used was the NAG [39] D02NHF routine whereas Verwer et al. used the more sophisticated DASSL [43] code. It would be unwise, however, to dismiss Method II on the basis of the present results without first testing its performance using a more sophisticated DAE/ODE solver.

Method III avoids the difficulties encountered when the node spacing becomes too close, by incorporating the minimum node spacing as a parameter in the method. Ultimately the choice of this parameter is somewhat problem specific and is therefore contrary to the need for versatility but it does however succeed in preserving a high degree of robustness in the implementation. Present day ODE solvers, including those used in the numerical experiments, possess such a parameter for the time integration, so it is perhaps quite acceptable to have a similar parameter associated with the spatial discretisation.

The results obtained in the last section using Methods I, II and III to solve Problems 1, 2, 3 and 4 can be qualitatively summarised as follows.

	Robustness	Efficiency	Accuracy
Method I	2	3	1
Method II	3	1	3
Method III	1	2	2

The three algorithms are numbered in order of merit, 1 indicating the best performance and 3 indicating the worst performance. Method III was the most robust algorithm because, for the default choice of parameters, the method never failed. Methods I and II, on the other hand, failed for a small number of implementations. Method II was the most efficient algorithm but unfortunately was also the most inaccurate. On the whole, Method III behaved reasonable efficiently and accurately compared to the other algorithms.

The comparisons made with the conventional fixed grid MOL approach (Method I) suggest that both the adaptive mesh strategies (Methods II and III) are only really viable for quite difficult parabolic problems since the computing overhead in the calculation of the grid is significant. For those problems which are only mildly difficult, such as Problem 1, the adaptive mesh strategies are a lot less effective than the conventional fixed grid approach. Their great advantage, however, is their ability to effectively solve difficult parabolic problems, such as Problem 2, and to a lesser extent Problems 3 and 4, where the conventional approach performs poorly.

Chapter 7

Conclusions

The objective of all efficient numerical methods could be described as the discrete approximation of a problem with uniform accuracy over its entire domain. In the context of one-dimensional parabolic partial differential equations the domain of interest typically involves finite intervals of space and time. For these problems the above objective has not yet been fully realised.

The standard approach to the numerical solution of parabolic equations was described in Chapter 3. Here, spatial discretisation leads to a system of (usually stiff) ODEs which are then solved routinely using existing high quality ODE integrators. This constitutes the conventional Method of Lines (MOL) approach. The great popularity of the MOL approach stems from its ability to reduce even the most diverse initial/boundary-value problems in parabolic partial differential equations to the familiar initial-value problem for a system of ordinary differential equations. This ability has made the MOL approach the single most versatile method for the numerical solution of parabolic equations. In this area versatility is hard to achieve due to the great diversity of problems yet the MOL approach stands out as an effective method where few others exist.

Since the advent of automatic temporal integration methods (typified by the Gear codes [21,22]) the efficient solution of ODE systems has become more or less routine. Today, one could say that the goal of "error equidistribution" has actually been achieved for the temporal dimension of parabolic problems. What remains unrealised is the equidistribution of errors in the spatial domain of these problems. The fact is clearly illustrated by the uniform mesh implementations at the end of Chapter 3.

Chapter 4 dealt with the analysis of so-called adaptive mesh strategies. The aim of adaptive spatial meshing is to provide efficiency in the spatial dimension hitherto only possible in the temporal integration. The apparent diversity of strategies in a sense mirrors the underlying complexity of parabolic problems. At a more fundamental level, however, adaptive mesh strategies differ in only one way, namely, the manner in which they correlate the solution and the numerical grid. Characterisation of adaptive mesh strategies on the basis of this property yields two main classes: local mesh refinement methods and mesh moving methods. In the first class, the solution and grid are loosely related with each being determined in different ways. This leads to approaches which tend to be heuristic in nature and are not very robust. Methods belonging to the second class treat both calculations in a very homogeneous way. The grid evolution is controlled in a more formal manner resulting in implementations which are robust and free of heuristics. A considerable effort, however, is devoted to calculating the grid. Present research in this field suggests that the mesh moving methods provide a more viable approach towards adaptive meshing than the local mesh refinement methods.

Given the apparent superiority of the mesh moving approach it is not surprising that the two algorithms chosen for further study in Chapter 5 belong to this class. The fact that both algorithms involve MOL type implementations is an important factor because strategies such as these lend themselves easily to incorporation into existing solution methods.

The first algorithm, from Verwer et al. [53], is particularly straightforward to implement as it is based upon existing Finite Difference methods. The method possesses an interesting temporal grid smoothing property which retains efficiency in the time integration. Few other adaptive mesh strategies address the closely related problems of spatial and temporal resolution in this way. Control of the adaptive meshing process is exercised via a single parameter which has a clear physical significance. This leads to straightforward implementations.

The second algorithm, Moving Finite elements [36,35], is a generalisation of the conventional (fixed grid) Galerkin Finite Element method. This approach is geared solely towards the goal of spatial resolution without regard to the time integration process. Three parameters determine the behaviour of the adaptive meshing process but only one of these has any direct physical significance. Consequently, although highly accurate, implementations tend to be non-optimal in terms of computational expense.

In general, the first algorithm is computationally inexpensive to implement, but tends to be inaccurate, whereas the second method is relatively expensive to implement, but retains a high level of accuracy and robustness. The robustness of the first method was called into question when, in the case of Problem 2, fatal convergence errors occurred for small values of the temporal grid smoothing parameter. In such cases, the semidiscretisation produces a system of Differential-Algebraic equations (DAEs). However, these systems are very complex and require more sophisticated temporal integrators than those used in the present experiments.

Comparisons with the conventional fixed grid MOL approach demonstrated the effectiveness of the adaptive mesh strategies in the solution of difficult parabolic problems. However, for problems of intermediate difficulty, both methods were only marginally better than the conventional approach. One disadvantage of the moving mesh approach is that considerable effort is devoted to calculating the grid. Clearly for difficult problems, where the solution is critically dependent on the spatial grid, this effort is very worthwhile. For less difficult problems, on the other hand, expensive calculation of a non-critical grid constitutes a waste of effort. The adaptive mesh strategies implemented here are therefore suited to the solution of **difficult** problems. For the easy to mildly-difficult problems the conventional MOL approach is still the preferred method. Current research is still mainly concerned with the solution of difficult problems and the improvement of existing algorithms. The trend appears to be moving towards the use of adaptive mesh strategies which combine the recognised advantages of both the local mesh refinement and mesh moving approaches. See for example Adjerid and Flaherty [1] and Verwer et al. [54].

Looking to the future, it is likely that adaptive mesh algorithms will soon be available as options within standard packages. Eventually, the automatic choice between fixed or adaptive strategies will also become possible, in the same way that some present-day ODE solvers automatically select the stiff or non-stiff options in the temporal integration. The ultimate goal, however, would be the production of a single adaptive mesh algorithm which reduces to the conventional fixed grid approach, whenever appropriate. Such a strategy would then provide efficient solutions over the entire range of parabolic problems.

Whatever the outcome of present research in the field of adaptive meshing there is little doubt that the adaptive integration of parabolic equations in the spatial dimension, will soon be routinely possible. Thus, the objective of evenly distributing the numerical errors over the entire problem domain will have been realised. The analysis presented in this thesis clearly indicates that the adaptive mesh strategies currently being developed, go a long way towards realising this objective.

Bibliography

[1] S. Adjerid, J.E. Flaherty, *A moving finite element method with error estimation and refinement for one-dimensional time-dependent partial differential equations*, SIAM J. Numer. Anal., 23 (1986), pp. 778-796.

[2] C.M. Ablow, S. Schechter, *Campylotropic coordinates*, J. Comput. Phys. 27, 351-362 (1978).

[3] J.G. Blom, J.M. Sanz-Serna, J.G. Verwer, *On simple moving grid methods for one-dimensional evolutionary partial differential equations*, J. Comput. Phys. 74, 191-213 (1988).

[4] R. Braddock, J. Noye, *Generalised variable grid size methods with application to the diffusion equation*, in *Computational Techniques and Applications: CTAC-83* (Noye and Fletcher, eds.), Elsevier Science Publishers B.V. (North-Holland) 1984.

[5] T.D. Bui, A.K. Oppenheim, D.T. Pratt, *Recent advances in methods for numerical solution of ODE initial value problems*, J. Comp. Appls. Maths. 11 (1984), 285-296.

[6] G.D. Byrne, A.C. Hindmarsh, *A polyalgorithm for the numerical solution of ordinary differential equations*, ACM Trans. Math. Soft., Vol. 1, No. 1, 71(1975).

[7] G.F. Carrier, K.E. Pearson, *Partial differential equations*, Academic Press 1976.

[8] T.H. Chong, *A variable mesh finite difference method for solving a class of parabolic differential equations in one space variable*, SIAM J. Numer. Anal., Vol. 15, No. 4, August 1978.

[9] M. Ciment, S.H. Leventhal, B.C. Weinberg, *The operator compact implicit method for parabolic problems*, J. Comput. Phys. 28, 135-166 (1978).

[10] J.D. Cole, *On a quasi-linear parabolic equation occurring in aerodynamics*, Quart. Appl. Math. 9, 225-236 (1951).

[11] J.M. Coyle, J.E. Flaherty, R. Ludwig, *On the stability of mesh equidistribution strategies for time dependent partial differential equations*, J. Comput. Phys. 62, 26-39 (1986).

[12] C.F. Curtiss, J.O. Hirschfelder, Proc. Nat. Acad. Sci. U.S.A. 38, 235 (1952).

[13] G. Dahlquist, *A special stability problem for linear multistep methods*, BIT 3, 27-43 (1963).

[14] S.F. Davis, J.E. Flaherty, *An adaptive finite element method for initial-boundary value problems for partial differential equations*, SIAM J. Sci. Stat. Comp., 3 (1982), pp. 6-27.

[15] P.M. Dew, J.E. Walsh, *A set of library routines for solving parabolic equations in one space variable*, ACM Trans. Math. Soft., Vol. 7, No. 3, Sept 1981, pages 295-314.

[16] E.A. Dorfi, L. O'C Drury, *Simple adaptive grids for one-dimensional initial value problems*, J. Comput. Phys. 69, 175-195 (1987).

[17] D.J. Evans, A.R. Abdullah, *A new explicit method for the diffusion-convection equation*, Comp. and Maths. with Appls. Vol. 11, nos. 1-3, pp. 145-154, 1985.

[18] C.A.J. Fletcher, *Computational galerkin methods*, Springer series in Computational Physics, Springer-Verlag 1984.

[19] G.E. Forsythe, W. Wasow, *Finite difference methods for partial differential equations*, John Wiley and Sons (1960).

[20] P.R. Garabedian, *Partial differential equations*, John Wiley and Sons (1964).

[21] C.W. Gear, *The automatic integration of stiff ordinary differential equations*, in *Proc. of the 1968 IFIP Congress, Edinburgh, Scotland* (Morrel, ed.) North-Holland, Amsterdam, 1968, p. 187.

[22] C.W. Gear, *Numerical initial value problems in ordinary differential equations*, Prentice Hall 1971.

[23] R.J. Gelinas, S.K. Dos, K. Miller, *The moving finite element method: Applications to general partial differential equations with multiple large gradients*, J. Comput. Phys. 40, 202-249 (1981).

[24] A.C. Hindmarsh, ACM SIGNUM Newsletter 15, No. 410 (1980).

[25] A.C. Hindmarsh, *ODEPACK, A systematised collection of ODE solvers*, in *Scientific Computing* (Stepleman et al., eds.), North-Holland 1983, p. 55.

[26] A.C. Hindmarsh, G.D. Byrne, *Applications of EPISODE: An experimental package for the integration of systems of ordinary differential equations*, in *Numerical Methods for Differential Systems* (Lapidus and Schiesser, eds.), Academic Press (1976).

[27] T.R. Hopkins, R. Wait, *A comparison of galerkin, collocation and the method of lines for partial differential equations*, Int. J. Numer. Meths. Eng., Vol. 12, 1081-1107 (1978).

[28] H.B. Keller, *A new finite difference scheme for parabolic problems*, in *Numerical Solution of Partial Differential Equations (SYNSPADE 1970) Vol. 2* (Hubbard, ed.), Academic Press, New York, 1970.

[29] D.C.L. Lam, R.B. Simpson, *Centred differencing and the box scheme for diffusion-convection problems*, J. Comput. Phys. 22, 486-500 (1976).

[30] J.J. Jr. Lambiotte, R.G. Voigt, *The solution of tridiagonal linear systems on the CDC STAR-100 computer*, ACM Trans. Math. Soft., Vol. 1, No. 4, 308-329, Dec. 1975.

[31] M. Lentini, V. Pereyra, *An adaptive finite difference solver for nonlinear two-point boundary problems with mild boundary layers*, SIAM J. Numer. Anal., Vol. 14, No. 1, March 1977.

[32] N.K. Madsen, *Non-stiff adaptive moving node techniques,* in Scientific Computing (Stepeleman et al., eds.), IMACS/North-Holland, 1983.

[33] N.K. Madsen, R.F. Sincovec, *Algorithm 540, PDECOL: General collocation software for partial differential equations*, ACM Trans. Math. Soft., Vol. 5, No. 3, Sept 1979.

[34] K. Miller, *Alternate modes to control the nodes in the moving finite element method*, in *Adaptive Computational Methods for Partial Differential Equations* (Babuska, Chandra and Flaherty, eds.), SIAM, Philadelphia 1983.

[35] K. Miller, *Moving finite elements II*, SIAM J. Numer. Anal., Vol. 18, No. 6, Dec. 1981.

[36] K. Miller, R.N. Miller, *Moving finite elements I*, SIAM J. Numer. Anal., Vol. 18, No. 6, Dec. 1981.

[37] A.R. Mitchell, D.E. Griffiths, *The finite difference method in partial differential equations.* John Wiley and Sons (1980).

[38] A. R. Mitchell, R. Wait, *The finite element method in partial differential equations*, John Wiley and Sons (1977).

[39] Numerical Algorithms Group, *The NAG Fortran Library - Mark 12.*

[40] J. Noye, *Numerical solution of partial differential equations*, North-Holland 1982.

[41] C.E. Pearson, *On a differential equation of boundary layer type*, J. Math. Phys. 47 (1968), pp. 134-154.

[42] L.R. Petzold, *Automatic selection of methods for solving stiff and non-stiff systems of ordinary differential equations*, SIAM J. Sci. Stat. Comput. 4, 136(1983).

[43] L.R. Petzold, *A description of DASSL: A differential-algebraic system solver*, in Scientific Computing, R.S. Stepeleman et al. (eds.), North-Holland, Amsterdam, 1983, p. 65.

[44] R.B. Philips, M.E. Rose, *Compact finite difference schemes for mixed initial-boundary value problems*, SIAM J. Numer. Anal. 19, No. 4 (1982), pp. 698-720.

[45] W.L. Seward, G. Fairweather, R.L. Johnston, *A survey of higher-order methods for the numerical integration of semi-discrete parabolic problems*, IMA J. Numer. Anal. (1984) 4, 375-424.

[46] R.F. Sincovec, N.K. Madsen, *Software for nonlinear partial differential equations*, ACM Trans. Math. Soft., Vol 1., No. 3, Sept. 1975, pages 232-260.

[47] G.D. Smith, *Numerical solution of partial differential equations*, Oxford University Press 1965.

[48] M.D. Smooke, M.L. Koszykowski, *Fully adaptive solutions of one-dimensional mixed initial-boundary problems with applications to unstable problems in combustion*, SIAM J. Sci. Stat. Comput., Vol. 7, No. 1, January 1986.

[49] B.K. Swartz, *The construction and comparison of finite difference analogues of some finite element schemes*, in *Mathematical Aspects of Finite Elements in Partial Differential Equations* (De Boor, ed.), Academic Press, New York, 1974, pp. 279-312.

[50] J.F. Thompson, Z.U.A. Warsi, C.W. Mastin, *Numerical grid generation: Foundations and applications*, Elsevier Science Publishers B.V. (North-Holland) 1985.

[51] J.F. Thompson, *A survey of dynamically adaptive grids in the numerical solution of partial differential equations*, Appl. Numer. Maths. 1 (1985), 3-27, North-Holland.

[52] J.M. Varah, *On the solution of block-tridiagonal systems arising from certain finite-difference equations*, Math. Comp. 26, No. 120, 859-868 (1972).

[53] J.G. Verwer, J.G. Blom, R.M. Furzeland, P.A. Zageling, *A moving-grid method for one-dimensional PDEs based on the method of lines*, Report NM-R88, Centre for Mathematics and Computer Science (CWI), Amsterdam 1988.

[54] J.G. Verwer, J.G. Blom, J.M. Sanz-Serna, *An adaptive moving-grid method for one-dimensional systems of partial differential equations*, Report NM-R8804, Centre for Mathematics and Computer Science (CWI), Amsterdam 1988.

[55] M. Vinokur, *On one-dimensional stretching functions for finite-difference calculations*, J. Comput. Phys. 50, 215-234 (1983).

[56] A.J. Wathen, M. Baines, *On the structure of the moving finite element equations*, IMA J. Numer. Anal. (1985) 5, 161-182.

[57] A.B. White Jr., *On the numerical solution of initial/boundary-value problems in one space dimension*, SIAM J. Numer. Anal., Vol. 19, No. 4, Aug. 1982.